NWC TP 6842

2

Reconstruction of Sea State One

by

David J. Keller
Fuze and Sensors Department

FEBRUARY 1988

NAVAL WEAPONS CENTER
CHINA LAKE, CA 93555-6001

Naval Weapons Center

FOREWORD

The objective of this study was to statistically reconstruct a surface representation of Sea State One based on direct measurement of a single point. The point measurement was obtained by a specialized piece of test equipment, the wave computer. The test was conducted at the Tower of the Naval Ocean Systems Center (NOSC), San Diego, Calif., approximately a mile off the coast of San Diego on 13 February 1985 under the direction of personnel of the Naval Weapons Center (NWC), China Lake, Calif.

The work was performed under SEATASK 62G-28706-008-1-S0167.

This report has been reviewed for technical accuracy by Joe McKenzie.

Approved by
C. L. SCHANIEL
Head, Fuze and Sensors Department
September 1987

Under authority of
J. A. BURT
Capt., U.S. Navy
Commander

Released for publication by
G. R. SCHIEFER
Technical Director

NWC Technical Publication 6842

Published by .... Technical Information Department
Collation .... Cover, 65 leaves
First printing .... 60 copies

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*) AD-A197280

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): NWC TP 6842	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Naval Weapons Center; 6b. OFFICE SYMBOL (If Applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code): China Lake, CA 93555-6001	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: NAVSEA; 8b. OFFICE SYMBOL (If Applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code): Washington, DC 20362-5101	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO.; PROJECT NO.; TASK NO. 62G-28706-008-1-S0167; WORK UNIT NO.

11. TITLE (*Include Security Classification*)

RECONSTRUCTION OF SEA STATE ONE

12. PERSONAL AUTHOR(S)

David J. Keller

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	From 85 Feb To 87 Sep	February 1988	128

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
08	03	
20	04	

18. SUBJECT TERMS (*Continue on reverse side if necessary and identify by block number*)

Mathematical modeling of sea surface
Wave computer system

19. ABSTRACT (*Continue on reverse side if necessary and identify by block number*)

(U) The objective of this study is to statically reconstruct a surface representative of Sea State One based on direct measurement of a single point. The point measurement was obtained by a specialized piece of test equipment known as the wave computer, which consists of a capacitive probe, laser, screen, and video camera, to give the elevation of the sea surface and a unit normal at a single point of observation. The computer programs for reconstructing a single point of the sea surface are included.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL: David J. Keller	22b. TELEPHONE (Include Area Code): (619) 939-1271; 22c. OFFICE SYMBOL: Code 3337

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted.
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

CONTENTS

ACKNOWLEDGMENT

The author, who was responsible for the conception, design, and data reduction of the wave computer system, realizes that the results of this effort would have been impossible without the tireless efforts of the following NWC personnel. Joe McKenzie coordinated the overall test plan and served as the project engineer. Don Meyer served as our mechanical engineer and solved the numerous structurally related problems associated with this effort. Bill Meyer designed and implemented the capacitive probe signal conditioner of the wave computer system. Lastly, a special thanks is extended to Wendell Peters who served as a fabrication and field test technician and who very effectively supported all of the foregoing tasks.

INTRODUCTION

The objective of this study is to statically reconstruct a surface representative of Sea State One based on direct measurement of a single point. This point measurement was obtained by a specialized piece of test equipment known as the wave computer. The wave computer by means of a capacitive probe, laser, screen, and video camera provides the elevation of the sea surface and a unit normal at a single point of observation. This test was conducted at the Naval Ocean Systems Center (NOSC) Tower* about a mile off the coast of San Diego on 13 February 1985 under the direction of personnel from the Naval Weapons Center, China Lake, Calif.

THE WAVE COMPUTER SYSTEM

In this section only a general overview of the wave computer system will be offered. A more comprehensive treatment of this subject is available in Appendix A, which also includes a system circuit schematic. The wave computer system was deployed off the south side of the NOSC Tower as shown in Figure 1.

Figure 2 is a block diagram of the wave computer system. The elevation of the sea surface is reported by a capacitive probe subsystem. A capacitive probe consists of an insulated wire using the seawater as the outer conductor, thus forming a coaxial cable whose capacitance per unit length is well defined. This capacitance, which is a function of the elevation of the sea surface, is then used to time a monostable one-shot. Thus, the output pulse duration of the one-shot is dependent on the elevation of the sea surface. These pulses are then integrated over several hundred cycles resulting in a DC level proportional to the elevation of the sea surface. The frequency response of the capacitive probe system is 120 hertz, and resolution down to 0.5 centimeter has been demonstrated. This elevation signal was labeled the 'Z vector' and is stored on the instrumentation tape recorder simultaneously with directional signals.

The directional information is provided by the beam screen subsystem. This subsystem consists of reflecting a parallel-with-gravity incident laser beam off the sea surface and observing its point of impact on a diffuse translucent screen. This observation is performed in real time at a sampling frequency of 60 hertz by a charge-injection-device (CID) video camera pointed at the diffuse screen. The composite video signal is then decoded by the wave computer central processor into x' and y' beam screen coordinates.

*The NOSC Tower was taken over by the Scripps Institute of Technology, University of California, San Diego, on 1 November 1986.

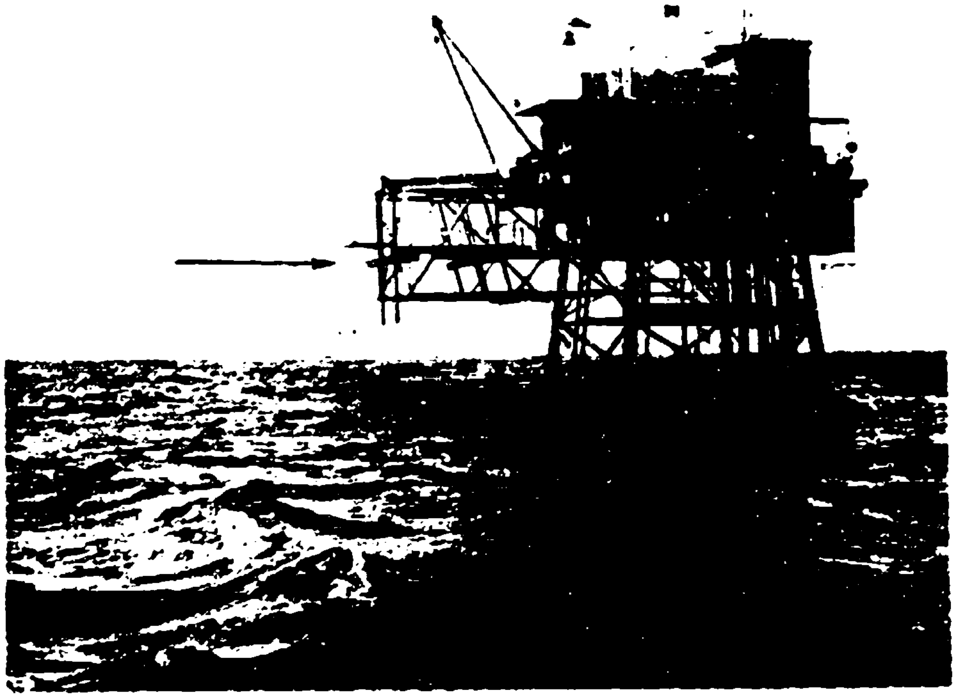


FIGURE 1. Wave Computer Beam Screen.

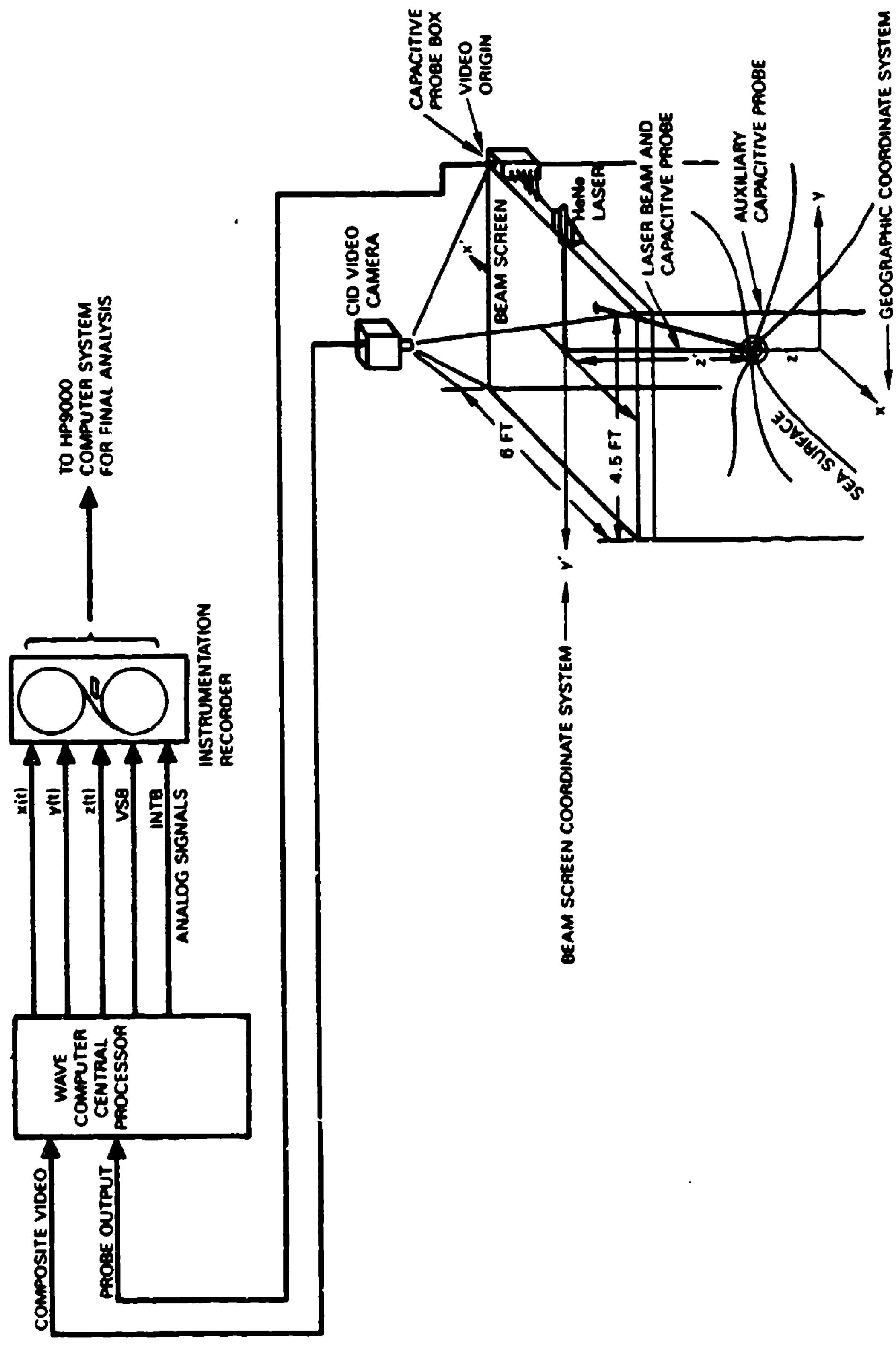

COMPOSITE VIDEO
PROBE OUTPUT
WAVE COMPUTER CENTRAL PROCESSOR
x(t)
y(t)
z(t)
VSB
INTB
ANALOG SIGNALS
INSTRUMENTATION RECORDER
TO HP9000 COMPUTER SYSTEM FOR FINAL ANALYSIS
CID VIDEO CAMERA
CAPACITIVE PROBE BOX
VIDEO ORIGIN
BEAM SCREEN
x'
HeNe LASER
LASER BEAM AND CAPACITIVE PROBE
AUXILIARY CAPACITIVE PROBE
y
z'
z
x
6 FT
4.5 FT
y'
SEA SURFACE
BEAM SCREEN COORDINATE SYSTEM
GEOGRAPHIC COORDINATE SYSTEM

FIGURE 2. Wave Computer System Block Diagram.

Thus, the wave computer system outputs the three data vectors in real time x', y', and z', which accurately represent the position and length of the laser beam reflected path. Moreover, the elevation of the sea surface is also accurately known.

GEOMETRIC INTERPRETATION OF WAVE COMPUTER OUTPUT

Figure 3 shows the beam screen geometry under analysis. Note that now we have a secondary coordinate system, that is, x, y, and z, which we shall refer to as "geographical coordinates." These geographical coordinates constitute the stationary reference system in which the sea surface is recorded and later reconstructed. The unit normal to the sea surface may now be obtained quickly in the geographical coordinate system by exploiting the first and second laws of reflection. That is, we know that the reflected and incident rays lie in the same plane and that the elevation angle of the unit normal is half that of the reflected laser beam.

With this information we may now write the elevation angle of the unit normal to the sea surface directly in terms of wave computer output as in Equation 1:

$$\phi(t) = (1/2)\tan^{-1}\left[\frac{\sqrt{[x'(t)]^2 + [y'(t)]^2}}{Z'(t)}\right] \qquad (1)$$

Moreover, by a considered choice of the geographical coordinates we may write the azimuthal angle of the unit normal in this coordinate system as in Equation 2:

$$\theta(t) = \tan^{-1}\left[\frac{y'(t)}{x'(t)}\right] - \pi \qquad (2)$$

With this information we may now write the unit normal in the geographical coordinate system as in Equation 3 (Reference 1):

$$\vec{N}(t) = \{\sin[\phi(t)\}\cos[\theta(t)]\}\vec{i} + \{\sin[\phi(t)]\sin[\theta(t)]\}\vec{j} + \cos[\phi(t)]\vec{k} \qquad (3)$$

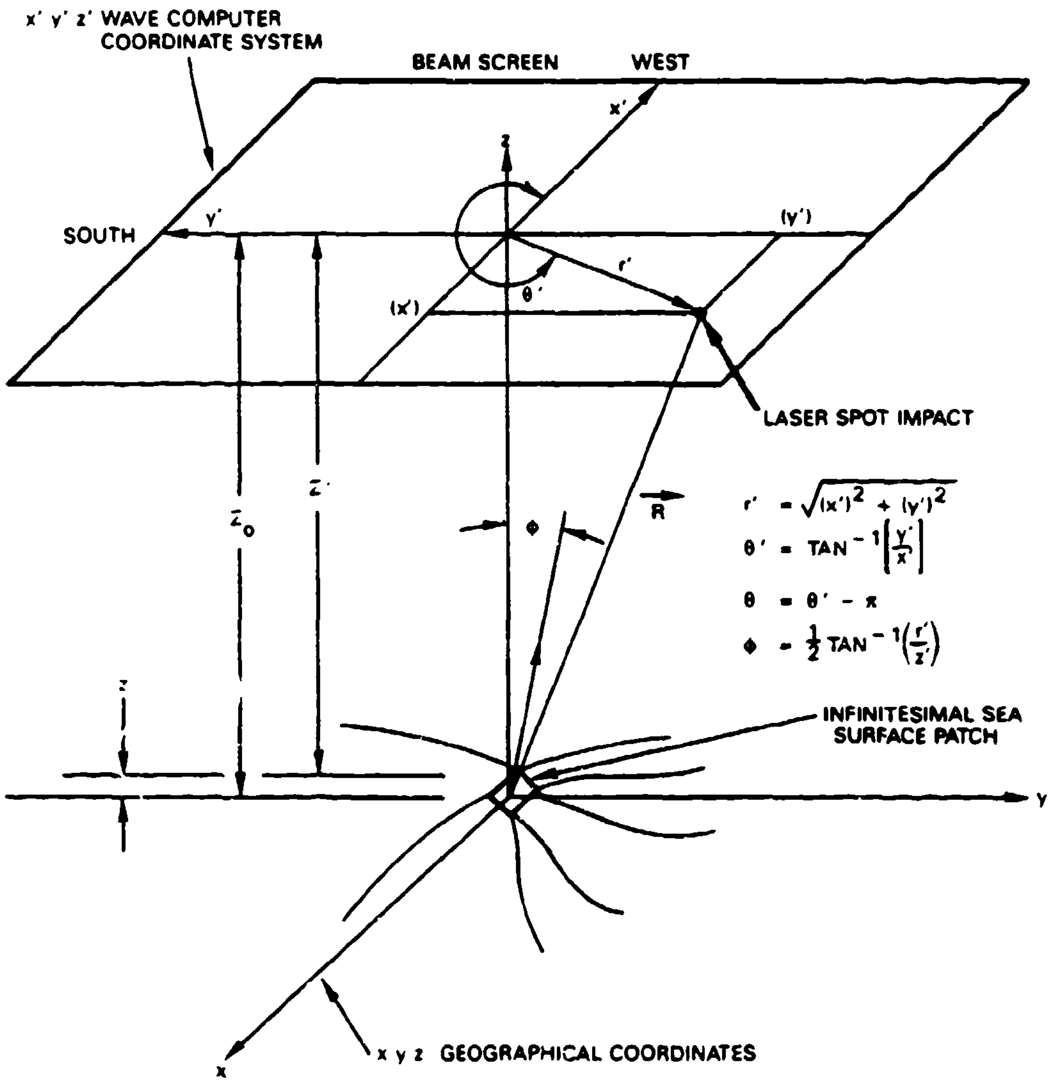

x' y' z' WAVE COMPUTER COORDINATE SYSTEM
BEAM SCREEN
WEST
x'
z
SOUTH
y'
(y')
r'
θ'
(x')
LASER SPOT IMPACT
z'
z_o
Φ
$\vec{R}$
$r' = \sqrt{(x')^2 + (y')^2}$
$\theta' = \text{TAN}^{-1}\left[\frac{y'}{x'}\right]$
$\theta = \theta' - \pi$
$\Phi = \frac{1}{2}\text{TAN}^{-1}\left(\frac{r'}{z'}\right)$
z
INFINITESIMAL SEA SURFACE PATCH
y
x
x y z GEOGRAPHICAL COORDINATES

FIGURE 3. Wave Computer Geometry Under Analysis.

For the geographical coordinate system "Z" dimension we have the simple relation in Equation 4:

$$Z(t) = Z_o - Z'(t) \tag{4}$$

where Z_o is chosen to eliminate zero frequency offset bias. It will be shown that the information contained in Equations 3 and 4 completely characterize the sea surface when observed over adequate periods of time.

The spatial partial derivatives of the sea surface in the x and y directions become immediately apparent when one considers an alternative form of the unit normal equation that is given by Equation 5 (Reference 2):

$$N(t) = \frac{-(Zx)\vec{i} - (Zy)\vec{j} + (1)\bar{k}}{\sqrt{(Zx)^2 + (Zy)^2 + 1}} \tag{5}$$

By comparing the form of Equation 5 to that of Equation 3 and equating coefficients of like unit vectors we obtain Equations 6 and 7:

$$\frac{dZ(t)}{dx} = -\tan[\phi(t)]\cos[\theta(t)] = Zx(t) \tag{6}$$

$$\frac{dZ(t)}{dy} = -\tan[\phi(t)]\sin[\theta(t)] = Zy(t) \tag{7}$$

As we shall see later the Fourier transform of these spatial derivatives will be indispensable in determining the wavelengths of the wave components.

MATHEMATICAL MODELING OF THE SEA SURFACE

We are now prepared to introduce a three-dimensional model of the sea surface. However, before we proceed, an outline of the fundamental assumptions is in order. First, the principle of time invariance of wave component amplitude, velocity and wavelength has been assumed. Without this assumption we could be lost in a maze of complexity. Second, the principle of spectral spatial invariance is assumed. That is, if we were to perform a wave computer measurement at a different location within a reasonable distance we should obtain the same spectral results upon applying a Fourier transform. Last, it is also assumed that each individual frequency component has associated with it a unique direction, velocity, and wavelength. With these assumptions put forth we may write the sea surface equation as a linear superposition of plane waves as indicated below in Equation 8:

$$Z(\overline{r},t) = \sum_{n=0}^{N-1} C_n \cdot \cos\left[\frac{2\pi}{\lambda_n}(\overline{r}_n \cdot \overline{U}_n - V_n t) - \psi_n\right] \tag{8}$$

where

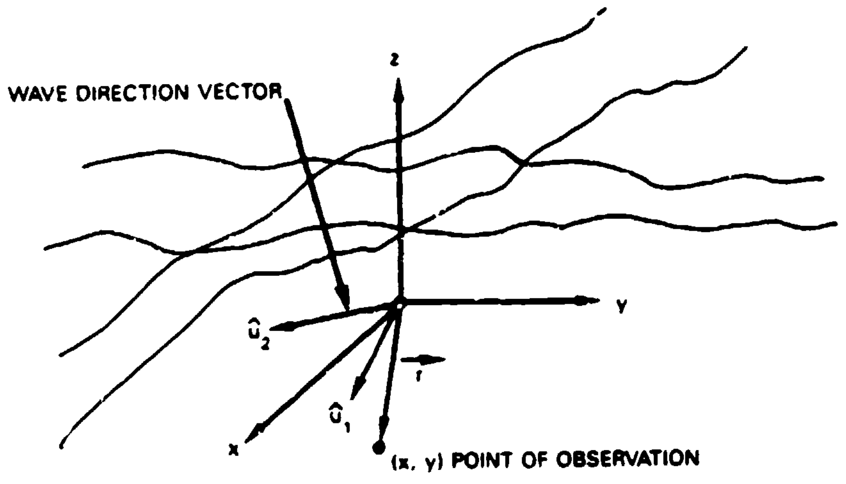
$\overline{r} \cdot \overline{U}_n$ = $x\cos(\gamma_n) + y\sin(\gamma_n)$ = projection on wave axis n
r = a 2-dimensional vector in the xy plane indicating position
t = time, seconds
N = number of wave components considered
C_n = amplitude coefficient of wave component
U_n = unit vector in the direction of wave component travel
V_n = wave component velocity, ft/sec
λ_n = wavelength of wave component, ft
ψ_n = angular phase delay of wave component, radians
γ_n = angular bearing of wave component propagation axis

Note that the dot product in the argument of the cosine term determines the projection of the point of observation "r" on the axis of propagation of the plane wave component under consideration. This operation is illustrated in Figure 4. If we were to evaluate Equation 8 at origin, which is the only point we are really observing, we would obtain Equation 9:

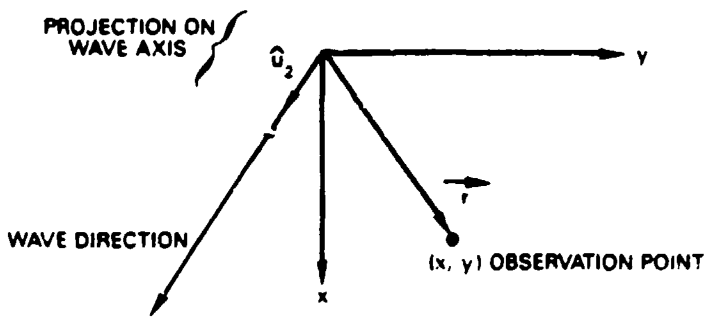
$$Z_o(t) = Z(\overline{o},t) = \sum_{n=0}^{N-1} C_n \cos\left[\left(\frac{2\pi V_n t}{\lambda_n}\right) + \psi_n\right] = \sum_{n=0}^{N-1} C_n \cos(W_n t + \psi_n) \tag{9}$$

where

$$W_n = \frac{2\pi V_n t}{\lambda_n} = \text{angular velocity}$$



(a) Projections on plane wave axes.



(b) Top view of plane vectors.

FIGURE 4. Plane Wave Components.

Taking the spatial derivatives of our sea surface model and again evaluating at origin yields Equations 10 and 11:

$$\frac{dZ_o(t)}{dx} = -2\pi \sum_{n=0}^{N-1} \left(\frac{C_n}{\lambda_n}\right) \cos(\gamma_n) \sin\left[\left(\frac{2\pi V_n t}{\lambda_n}\right) - \psi_n\right] \tag{10}$$

$$\frac{dZ_o(t)}{dy} = -2\pi \sum_{n=0}^{N-1} \left(\frac{C_n}{\lambda_n}\right) \sin(\gamma_n) \sin\left[\left(\frac{2\pi V_n t}{\lambda_n}\right) - \psi_n\right] \tag{11}$$

The spatial partial derivatives of our sea surface model in conjunction with Equations 6 and 7 suggest a means by which we may determine the wavelength of our plane wave components. This is, of course, if we can isolate the individual frequency components in our series model of the sea surface.

DATA REDUCTION OF WAVE COMPUTER SIGNALS

The wave computer data obtained from the NOSC Tower tests were stored on a Honeywell 101C instrumentation tape recorder. The temporally based analog data was then fed into the HP6942A multiprogrammer where the information was scaled and digitized. Once a complete data record was obtained, the HP9000 computer read the data from the multiprogrammer memory banks through the linking HPIB data bus. At this point the data was then stored on the HP7914 hard disk drive for later operations. The above operation was supervised by the HP9000 program "SEA_LINK." A diagram of the data reduction equipment layout is shown in Figure 5.

The next software operation involves converting the x'y'z' data files into angular formatted data files; that is, the structure of ϕ, θ, Z, in which the first two variables are the unit normal elevation and azimuthal angles. It should be noted that no information is lost or gained in this step; only a handier data format is obtained. This operation is performed by the program "ANGLER." We are now prepared to address the spectral aspects of the data-reduction process.

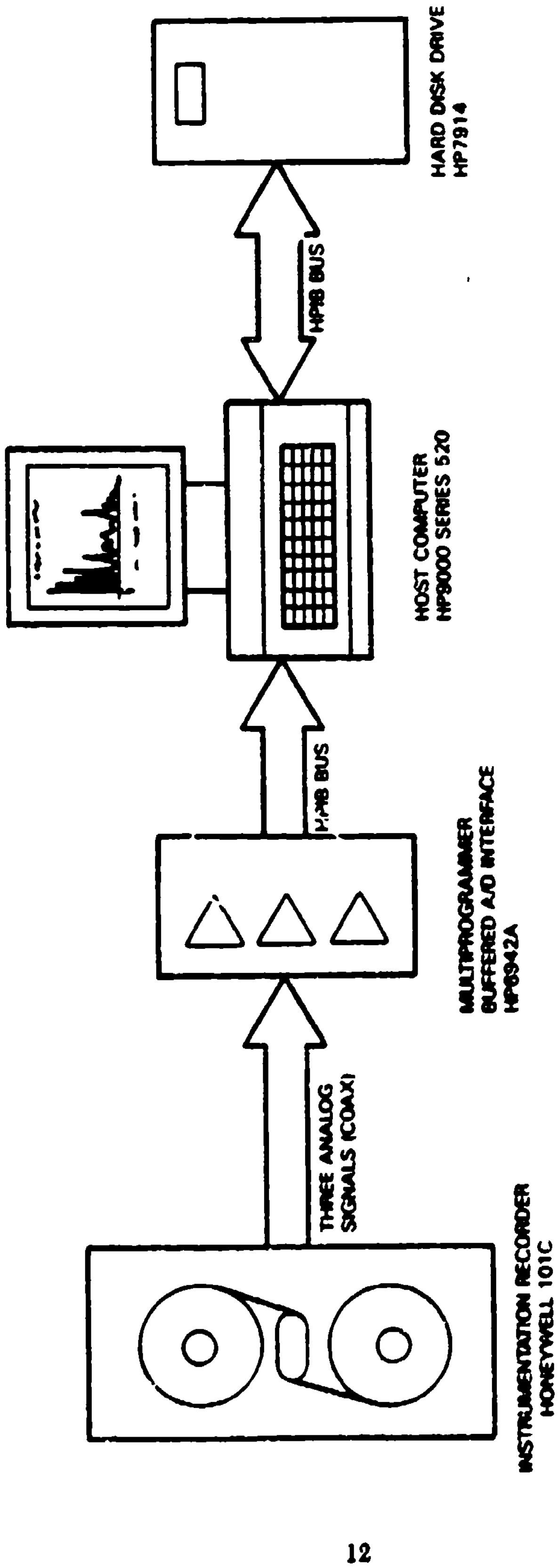


FIGURE 5. Layout of Data Reduction Equipment.

SPECTRAL ANALYSIS OF WAVE COMPUTER SIGNALS

The overall strategy of spectral analysis of the wave computer signals entails three operations. First, the Fourier transform of the "Z" vector is performed to provide us with the "C_n" coefficients of the sea surface model shown in Equation 8. This also provides information as to which particular frequency components are dominant in the spectrum. Second, we must determine the direction of propagation of the wave frequency component under consideration. Last, the wavelength associated with each wave frequency component must be determined, from which the wave velocity follows.

DETERMINATION OF THE AMPLITUDE SPECTRUM

The amplitude frequency spectrum of the sea surface follows directly from taking the Fourier transform of the "Z" vector. It has been shown in Appendix B that the Fourier transform of the "Z" vector results in the quantity in Equation 12:

$$F\{Z_o(t)\} = \int_0^{\infty} Z_o(t)e^{-j2\pi ft}\,dt = (P/2)C_k \text{sinc}[P(f - fw)] \tag{12}$$

Since in applicaton we are taking the Fast Fourier Transform (FFT) of the "Z" vector, our spectrum will differ by a proportionality constant equivalent to the quantized magnitude of the differential. Thus, we define the magnitude of the amplitude spectrum as in Equation 13:*

$$|\hat{Z}_o(k\Delta f)| = \left|\frac{F\{Z_o(t)\}}{Ts}\right| = \left|\sum_{n=0}^{N_p-1} Z_o(nTs)e^{-j2\pi nk/N_p}\right| \tag{13}$$

*Note: $\hat{Z}_o(k\Delta f)$ is the equivalent discrete Fourier transform spectrum variable of $Z_o(t)$ and should not be confused with it.

$$|\hat{Z}_o(k\Delta f)| = \left|\left(\frac{N_p}{P}\right)\left(\frac{P}{2}\right) C_k \operatorname{sinc}(P(k\Delta f - fw))\right| = \frac{N_p C_k}{2} \left|\operatorname{sinc}(P(k\Delta f - fw))\right| \quad (14)$$

where

P =	temporal length of data record, seconds
N_p =	number of FFT points (power of two)
Ts =	temporal sampling interval, seconds
fw =	actual frequency of wave component, hertz
Δf =	$1/N_p Ts$ quantized frequency step of FFT

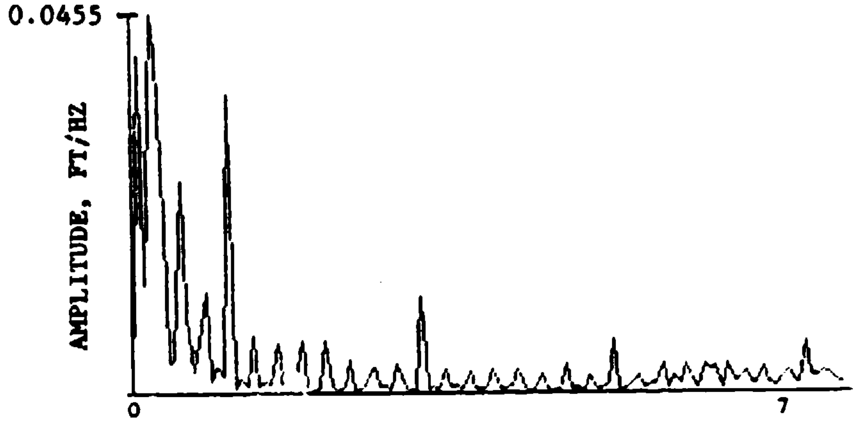
Assuming our frequency spacing to be of sufficient resolution we may then neglect the sinc term in that it will tend toward unity. We are left with the compact relation in Equation 15:

$$\left|\hat{Z}_o(k\Delta f)\right| = \frac{N_p C_k}{2} \quad (15)$$

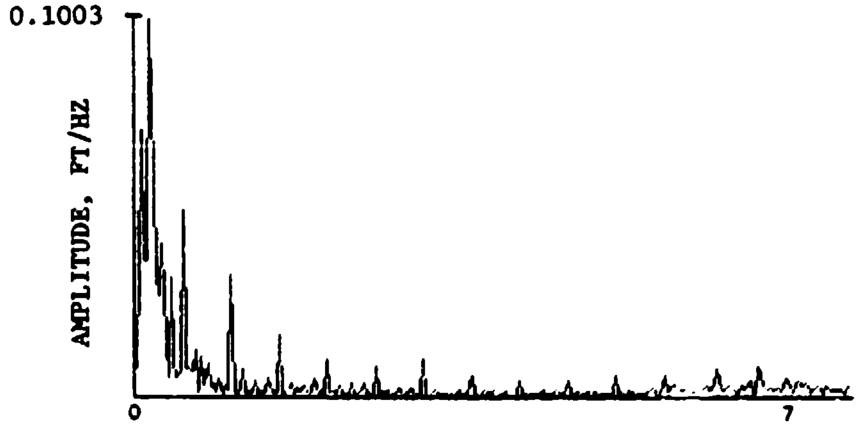
Thus the FFT operation performed on the "Z" vector by the program "Z_SPECTRUM" provides us with the first piece of the puzzle. Referring to Figures 6(a) through 6(c) we find the amplitude spectrum of the "Z" vector for sampling windows of 17, 34, and 60 seconds. We have only shown and used frequency components as far out as 7 hertz. Much beyond this point the spectrum becomes dominated by noise and unreliable even though the amplitude is quite minimal. It is interesting to note that the higher frequency components become less dominant with increasing temporal record lengths, whereas the lower frequency components seem to be quite stable. That is, the higher frequency wind wave components tend to demonstrate a random canceling effect while the lower frequency swell components appear quite stationary. This effect is also demonstrated in Figures 7(a) through 7(c) where a blowup of the amplitude spectrum origin is shown.

DETERMINATION OF WAVE DIRECTION

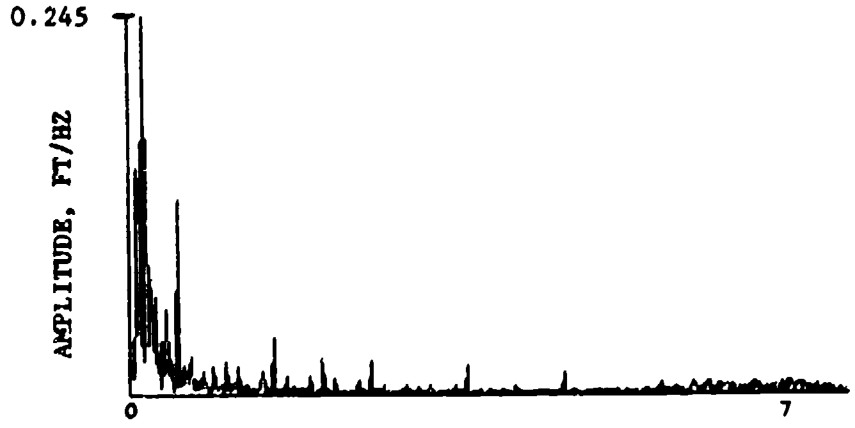
The direction of the plane wave fronts is of vital importance if the sea surface is ever to be reconstructed. To determine this parameter we have elected to discard mathematical rigor in favor of a more intuitive algorithm. Consider the top view of the beam screen shown in Figure 8. Note that the spot shown is not the laser spot but instead the imaginary point of impact of the unit normal to the sea surface. Further, consider



(a) 17-second data record.

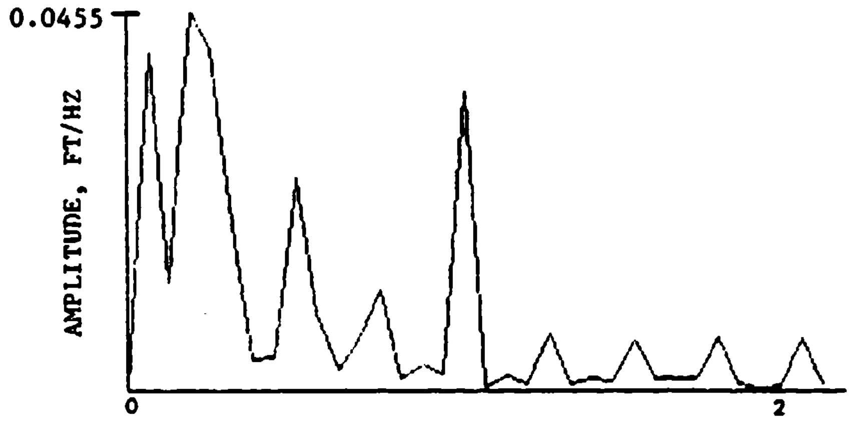


(b) 34-second data record.

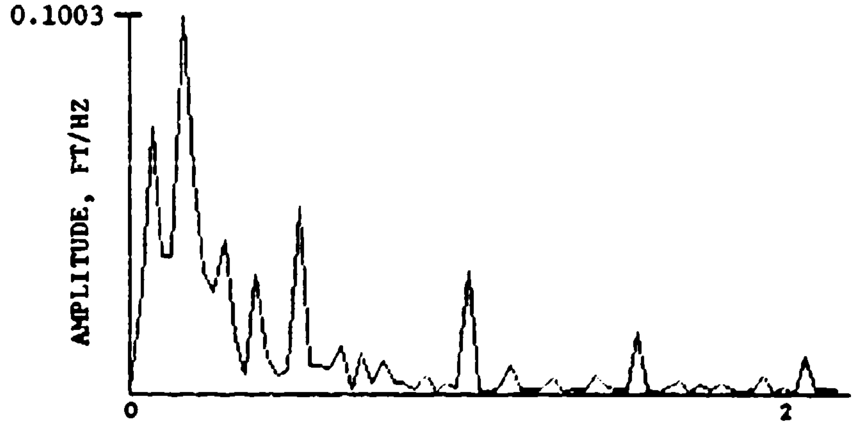


(c) 60-second data record.

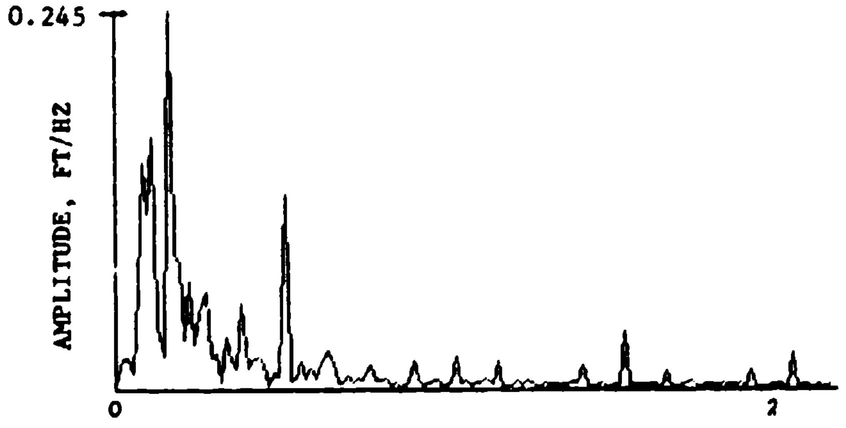
FIGURE 6. Z Vector Amplitude Spectrum.



(a) 17-second data record.



(b) 34-second data record.



(c) 60-second data record.

FIGURE 7. Truncated Z Vector Amplitude Spectrum.

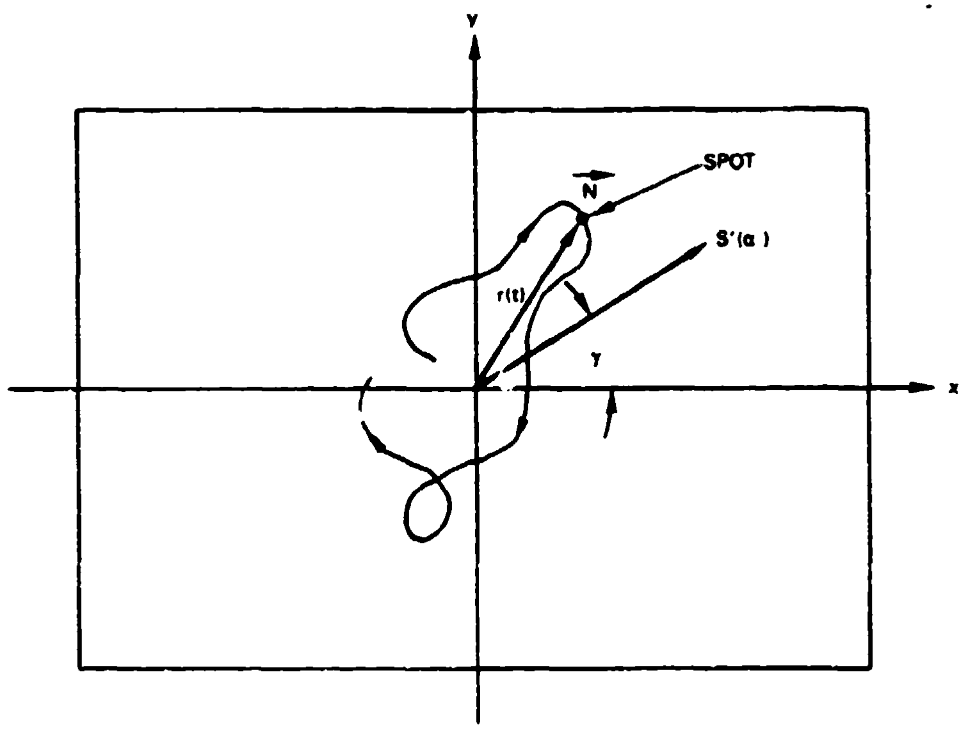


FIGURE 8. Top View of Beam Screen.

an arbitrary two-dimensional unit vector that may point in directions ranging from plus to minus 90 degrees, which we shall refer to as "S(α)", the direction sense vector. If we study the projection of the unit normal on the direction sense vector by performing a dot product operation, we could examine just how much angular displacement activity occurs along that direction. Moreover, if we take the Fourier transform of this projection and then for a fixed frequency, rotate the direction sense vector until a maxima is obtained, we would then obtain the direction of that wave component. This operation may be written in mathematical terms as Equation 16:

$$|G(f,\alpha)| = \left| \int_0^\infty \overline{N}(t) \cdot \overline{S}(\alpha) e^{-j2\pi ft} dt \right| \qquad (16)$$

where

$$\overline{N}(t) = \text{Unit Normal to the sea surface}$$
$$\overline{S}(\alpha) = \cos(\alpha)\overline{i} + \sin(\alpha)\overline{j} + (0)\overline{k}, \text{ directional test vector} \qquad (17)$$
$$\alpha = \text{bearing of directional test vector (radians)}$$

Expanding the dot product with the aid of Equation 3 we obtain Equation 18:

$$|G(f,\alpha)| = \int_0^P \{\sin(\phi)[\cos(\theta)\cos(\alpha) + \sin(\theta)\sin(\alpha)]\} e^{-j2\pi ft} dt \qquad (18)$$

or,

$$|G(f,\alpha)| = \left| \int_0^P \sin[\phi(t)]\cos[\theta(t) - \alpha] e^{-j2\pi ft} dt \right| \qquad (19)$$

The direction of the wave is then given by maximizing this integral by varying α for a fixed value of temporal frequency "f," which may be written as Equation 20:

$$\gamma(f) = \max_{\substack{\text{wrt } \alpha \\ \text{fix } f}} \left| \int_0^P \sin[\phi(t)]\cos[\theta(t) - \alpha] e^{-j2\pi ft} dt \right| \qquad (20)$$

It is conceded that the foregoing derivation is based more in intuition rather than mathematical rigor. However, the development of this algorithm was based on the observation of numerous directional spectrums. Note that several directional spectrums were computed in 5-degree directional steps for all three data record lengths and that multiple peaking was not observed. That is, for each frequency, a well defined cosine peak exists. Thus, it is logical to assume that the plane wave front is coming from the direction of maximum frequency activity. It is however, important to note that this algorithm is slope, not amplitude, sensitive in that we are transforming the sine of unit normal elevation angle. This operation is performed by the program "DIR_FFT" that writes the resulting wave directions to a data file. A printout of wave directions and frequencies is shown in Table 1 for a 17-second sampling window. The principal wave component directions are commensurate with observations recorded at the NOSC Tower.

DETERMINATION OF THE COMPONENT WAVELENGTHS

Determination of the frequency component wavelengths follows almost directly from the Fourier transform of the spatial derivatives. It has been shown in Appendix B that the magnitude of the Fourier transform of the first spatial derivative with respect to x results in Equation 22:

$$\left|\hat{Z}x(f)\right| = \frac{1}{T_s}\left|F\left\{\frac{\partial Z_o(t)}{\partial x}\right\}\right| = \left|\frac{-1}{T_s}\int_o^{\infty} \tan[\phi(t)]\,\cos[\theta(t)]e^{-j2\pi ft}dt\right| \quad (21)$$

$$|\hat{Z}x(f)| = \pi\left(\frac{N_pC_k}{\lambda_k}\right)|\cos(\gamma_k)\mathrm{sinc}(P(k\Delta f - fw))|\,^* \quad (22)$$

Similarly, the magnitude of the Fourier transform of the first spatial derivative with respect to y results in Equation 23:

$$|\hat{Z}y(f)| = \pi\left(\frac{N_pC_k}{\lambda_k}\right)|\sin(\gamma_k)\mathrm{sinc}(P(k\Delta f - fw))|\,^* \quad (23)$$

---

*The spectrums of both spatial derivatives are computed by the program "SPEC_DERIV."

TABLE 1. Bearings of Frequency Components.

Frequency, hertz	Bearing, degrees	Relative contribution, %
0.000	100.0	0.10
0.059	20.0	2.40
0.117	145.0	1.50
0.176	0.0	1.60
0.234	105.0	0.30
0.293	145.0	1.60
0.352	80.0	1.50
0.410	164.9	2.10
0.469	155.0	1.50
0.527	30.0	1.10
0.586	155.0	1.00
0.645	150.0	1.80
0.703	110.0	2.00
0.762	115.0	1.80
0.000	100.0	0.90
0.820	90.0	2.70
0.879	100.0	1.80
0.937	160.0	1.30
0.996	150.0	3.40
1.055	90.0	2.10
1.113	5.0	0.80
1.172	20.0	1.30
1.230	115.0	1.70
1.289	45.0	1.30
1.348	130.0	0.60
1.406	135.0	2.40
1.465	120.0	0.90
1.523	69.9	2.20
1.582	100.0	2.00
1.641	50.0	1.60
1.699	105.0	0.80
1.758	75.0	2.70
1.816	40.0	2.40
1.875	120.0	1.80
1.934	5.0	1.30
1.992	115.0	1.90
2.051	75.0	2.80
2.109	55.0	1.10
2.168	85.0	3.20
2.227	65.0	2.80
2.285	75.0	2.70
2.344	90.0	2.90
2.402	90.0	4.30
2.461	124.9	2.50
2.520	0.0	2.30
2.578	10.0	0.90
2.637	20.0	0.90
2.695	115.0	1.10
2.754	75.0	2.00
2.812	110.0	1.50
2.871	90.0	2.40
2.930	0.0	1.30
2.988	75.0	1.20
3.047	95.0	2.30
3.105	95.0	0.90
3.164	100.0	0.90

Now if we divide Equation 14 by Equation 23 and solve for the wavelength we obtain Equation 24:

$$\lambda(f) = 2\pi\left|\cos(\gamma)\right| \left|\frac{\hat{Z}_o(f)}{\hat{Z}x(f)}\right| = 2\pi\left|\sin(\gamma)\right|\left|\frac{\hat{Z}_o(f)}{\hat{Z}y(f)}\right| \tag{24}$$

Equation 24 provides the wavelength for each frequency component in the spectrum. The results of the above equations have been compared to known equilibrium wavelengths of gravity waves, that is Equation 25:

$$L = \frac{gT_p^2}{2\pi} = (5.12)T_p^2(\text{feet}) \tag{25}$$

where

T_p = wave period (seconds)

The results have been invariably in the correct order of magnitude for temporal frequencies in excess of 0.117 hertz. Below this frequency the deflection of the laser beam is less than 0.25 inch, the quantization step size of the wave computer central processor. Moreover, if the model adopted is to be regarded as consistent, by Equation 24 the following identity must bold true:

$$\left|\gamma(f)\right| = \tan^{-1}\left|\frac{Zy(f)}{Zx(f)}\right| \tag{26}$$

The angular quantity in Equation 26 was computed and compared to the iteratively determined wave direction given by Equation 20. For all frequencies below 3 hertz, Equation 26 provided the angular results of Equation 20 resolved to the first trigonometric quadrant.

With the wavelength known the wave component velocity follows simply by the relation given in Equation 27:

$$V_k = f_k\lambda_k \tag{27}$$

The operation of determining wave amplitude, wavelength, and velocity is performed by the program "MAKE_MODEL", which writes all relevant wave model data to a hard disk file for later use. A printout of sea surface modeling parameters is shown in Table 2 for the case of a 60-second data record.

RECONSTRUCTION OF THE SEA SURFACE

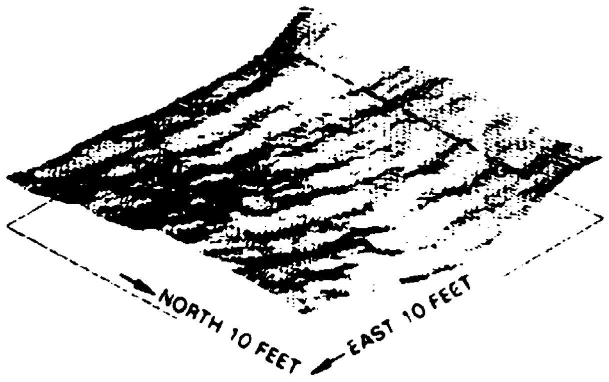
With all the necessary sea surface model coefficients determined, it is a simple matter to reconstruct the sea surface. Selecting arbitrarily a time t = 0, we need only evaluate the series model for all points in a square spatial coordinate system for as many frequency components as we wish to consider. This operation is performed by the program "MAKE_WAVES," which also allows the option of writing the sea surface to hard disk for later retrieval. Graphic display of the stored sea surface may be performed by the program "VIEW_SEA." All software used in this effort is included in Appendixes C through E. What follows is a presentation and discussion of several reconstructed sea surfaces of various spatial areas and data record lengths.

DISCUSSION OF RESULTS

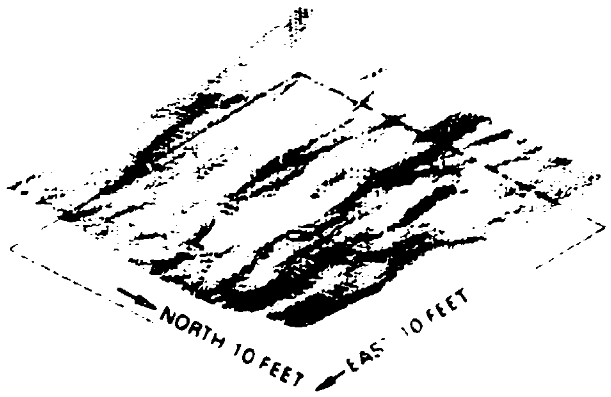
Figures 9(a) through (c) show a reconstruction of 100 square feet of sea surface based on data record lengths of 17, 34, and 60 seconds, respectively. These three images demonstrate the wave computer's remarkable ability to capture and reproduce detailed sea surface microstructure information. The accuracy and validity of the image definitely increases with longer data records, as evidenced by the views in Figures 10 and 11. It should be noted that the photograph in Figure 10 is not of the sea surface that we measured but rather of the sea early the next morning. It is, however, rather close based on spotlight observations while the test was in progress. The images shown in Figure 12(a) through (c) present an area of 625 square feet of reconstructed sea surface for various data record lengths. At this point in the spatial extension it becomes apparent that something is wrong with the 60-second data record reproduction shown in Figure 12(c). That is, there seems to be some peculiar stepping phenomena occurring along the far east-west axis. It is the author's opinion that this error is the result of shallow sloped wave quantization errors becoming apparent.

TABLE 2. Sea Surface Modeling Parameters (60-Second Record).

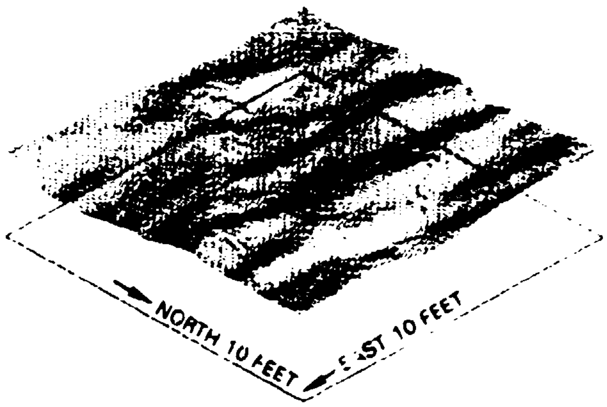
Frequency, hertz	Amplitude, feet	Phase, degrees	Bearing, degrees	Wavelength measured, ft	Wavelength theoretical, ft
0.000	2.02E-03	+180.0	+105.0	3.10E+01	9.99E+99
.015	3.53E-02	-101.0	+145.0	2.21E+01	2.39E+04
.029	3.97E-02	-12.0	+160.0	2.30E+01	5.97E+03
.044	2.02E-02	-133.4	-170.0	4.10E+01	2.65E+03
.059	1.14E-01	-170.0	-155.0	3.11E+01	1.49E+03
.073	2.59E-01	+44.4	-165.0	1.20E+02	9.54E+02
.088	2.01E-01	-53.4	-170.0	8.57E+01	6.63E+02
.103	2.08E-01	-164.1	-160.0	1.24E+02	4.87E+02
.117	7.49E-02	-98.7	+130.0	4.20E+01	3.73E+02
.132	3.97E-02	+149.8	-180.0	7.19E+01	2.95E+02
.146	4.31E-01	+77.2	+160.0	1.60E+02	2.39E+02
.161	1.54E-01	+59.3	+170.0	7.96E+01	1.97E+02
.176	1.44E-01	+28.1	+170.0	8.61E+01	1.66E+02
.190	5.80E-02	-115.5	-150.0	3.79E+01	1.41E+02
.205	1.23E-01	-173.0	+175.0	9.80E+01	1.22E+02
.220	5.28E-02	+36.1	-170.0	3.70E+01	1.06E+02
.234	8.69E-03	-145.0	-140.0	7.69E+01	9.32E+01
.249	1.12E-01	+59.1	+135.0	9.54E+01	8.26E+01
.264	3.75E-02	+122.9	-100.0	1.47E+01	7.36E+01
.278	4.84E-02	-73.5	-130.0	2.76E+01	6.61E+01
.293	8.24E-03	+129.5	-150.0	6.76E+00	5.97E+01
.308	6.09E-02	-108.4	+95.0	3.24E+01	5.41E+01
.322	3.23E-02	-129.6	+105.0	1.76E+01	4.93E+01
.337	2.60E-02	+158.7	+145.0	2.68E+01	4.51E+01
.352	1.00E-01	+108.1	+175.0	7.79E+01	4.14E+01
.366	2.50E-02	-30.2	+130.0	1.44E+01	3.82E+01
.381	3.65E-02	-11.5	-95.0	4.06E+01	3.53E+01
.396	3.95E-02	-27.9	+115.0	4.43E+01	3.27E+01
.410	2.93E-02	-135.6	-140.0	2.07E+01	3.04E+01
.425	4.99E-03	+126.1	+95.0	8.05E+00	2.84E+01
.439	1.91E-02	-173.7	-135.0	2.39E+01	2.65E+01
.454	1.67E-02	+156.0	+145.0	1.39E+01	2.48E+01
.469	2.23E-01	-91.2	+145.0	2.50E+02	2.33E+01
.483	9.73E-03	-140.0	-135.0	9.86E+00	2.19E+01
.498	1.33E-02	+69.5	+165.0	2.08E+01	2.06E+01
.513	3.25E-02	+52.4	+95.0	3.04E+01	1.95E+01
.527	1.50E-02	+31.8	-120.0	2.33E+01	1.84E+01
.542	2.59E-02	-50.0	-135.0	4.63E+01	1.74E+01
.557	1.38E-02	-71.0	-175.0	1.63E+01	1.65E+01
.571	3.11E-02	-165.8	+145.0	4.14E+01	1.57E+01
.586	4.51E-02	+93.0	+165.0	3.13E+01	1.49E+01
.601	2.92E-02	+101.7	+160.0	1.97E+01	1.42E+01
.615	6.97E-03	-3.0	-170.0	6.09E+00	1.35E+01
.630	7.20E-03	+75.7	-95.0	1.89E+01	1.29E+01
.645	1.56E-02	+125.9	+120.0	9.30E+00	1.23E+01
.659	8.75E-03	-125.5	-160.0	5.67E+00	1.18E+01
.674	1.34E-02	-69.6	+120.0	7.72E+00	1.13E+01
.688	1.21E-02	+142.6	-145.0	1.77E+01	1.08E+01
.703	2.86E-02	-119.7	-130.0	2.06E+01	1.04E+01
.718	1.22E-02	+168.1	+165.0	1.20E+01	9.94E+00
.732	1.21E-02	+52.8	+150.0	6.06E+00	9.54E+00
.747	1.35E-02	-33.3	+110.0	5.62E+00	9.17E+00
.762	5.11E-03	-110.2	+170.0	3.91E+00	8.82E+00
.776	7.42E-03	-64.3	+125.0	3.91E+00	8.49E+00
.791	3.84E-03	-115.4	-180.0	3.16E+00	8.18E+00
.806	9.16E-03	-133.8	-135.0	1.14E+01	7.89E+00
.820	3.23E-02	+76.5	-110.0	1.36E+01	7.61E+00
.835	1.13E-02	+148.5	-140.0	7.30E+00	7.34E+00
.850	4.37E-03	-132.9	+140.0	2.12E+00	7.09E+00
.864	9.16E-03	+113.7	+135.0	4.46E+00	6.85E+00
.879	1.19E-02	-156.3	+105.0	1.19E+01	6.63E+00
.894	1.19E-02	-16.0	-165.0	9.74E+00	6.41E+00
.908	3.27E-03	-53.0	-100.0	1.79E+00	6.21E+00
.923	7.26E-03	-171.2	+130.0	7.85E+00	6.01E+00
.937	3.98E-02	-47.9	+95.0	2.93E+01	5.83E+00
.952	7.01E-03	-170.1	+110.0	6.42E+00	5.65E+00
.967	1.08E-02	+110.9	+150.0	6.03E+00	5.48E+00
.981	6.07E-03	+70.8	-170.0	3.43E+00	5.32E+00
.996	7.72E-03	-133.6	-100.0	4.94E+00	5.16E+00
1.011	7.92E-03	-39.0	-150.0	8.04E+00	5.01E+00



(a) NOSC tower wave computer data 2/13/85; reel +3 footage 1190; 17 seconds; maximum crest to trough depth is 0.6 foot; aspect angle is 70 degrees; z axis gain is 6.



(b) NOSC tower wave computer data 2/13/85; reel +3 footage 1190; 34 seconds; maximum crest to trough depth is 0.4 foot; aspect angle is 70 degrees; z axis gain is 6.



(c) NOSC tower wave computer data 2/13/85; reel +3 footage 1190; 60 seconds; maximum crest to trough depth is 0.4 foot; aspect angle is 70 degrees; z axis gain is 6.

FIGURE 9. Reconstructed Sea Surface, 100 Square Feet.

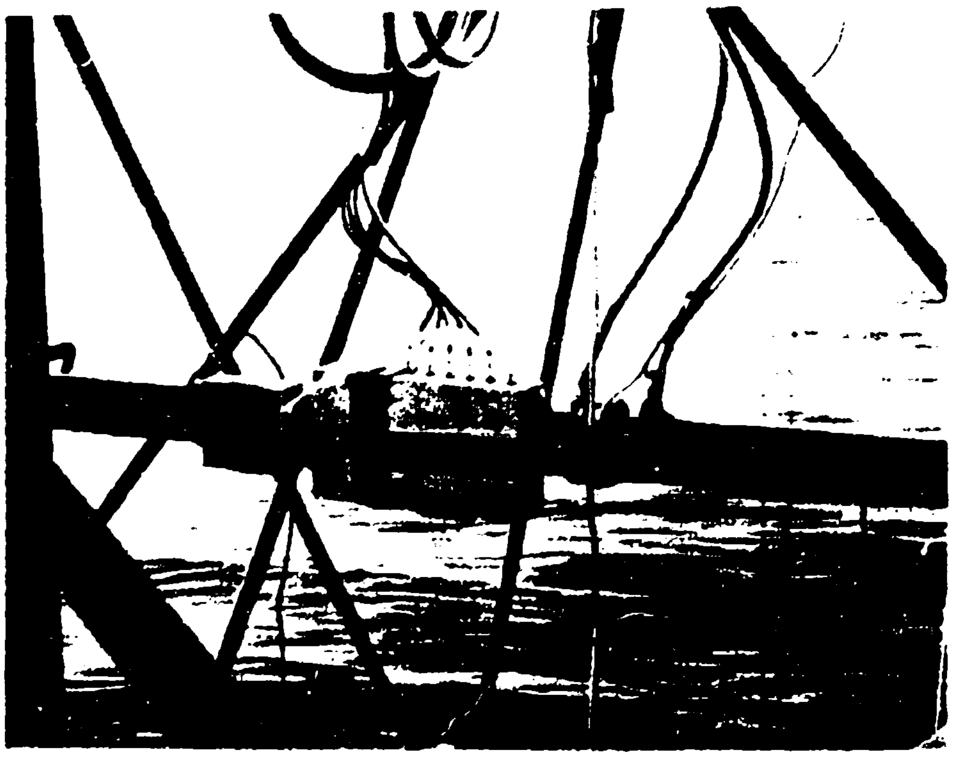


FIGURE 10. Southeastern View of Sea Surface From NOSC Tower Railing.

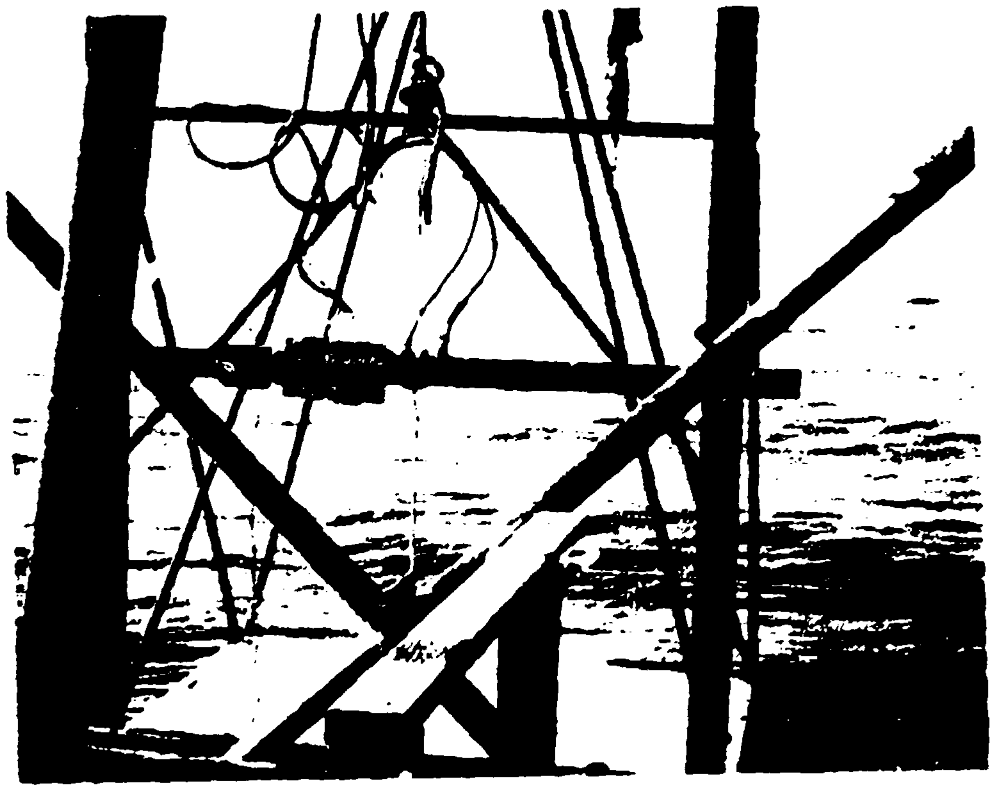
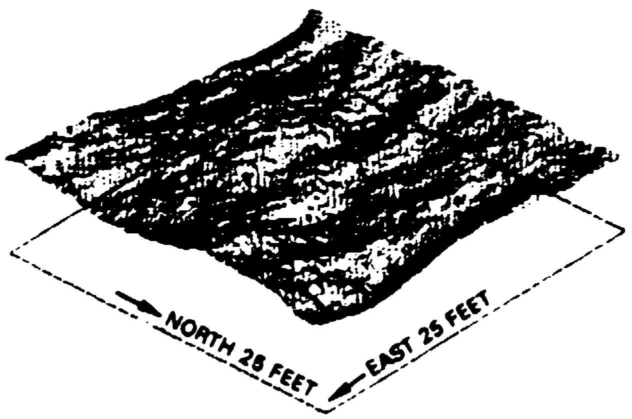
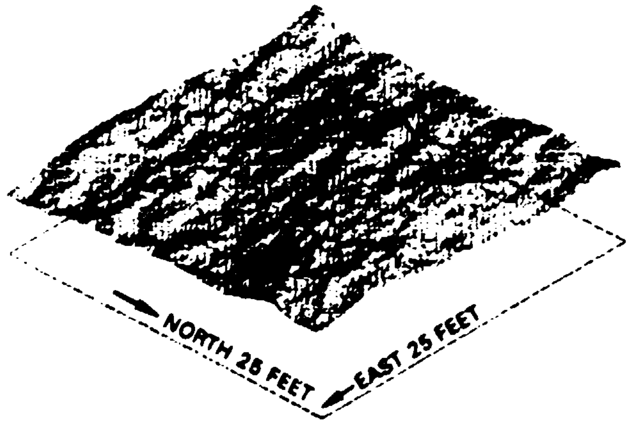


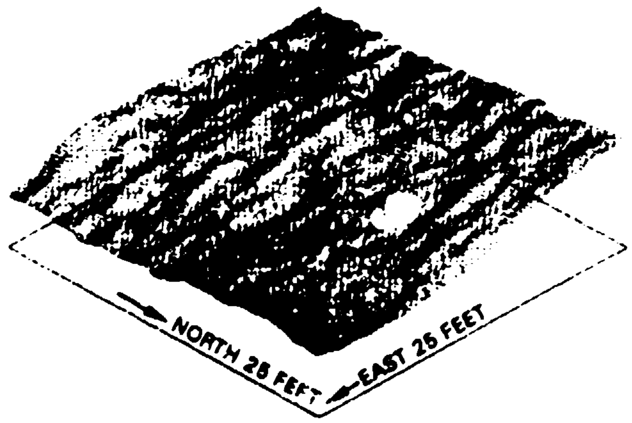
FIGURE 11. Southeastern View of Sea Surface From NOSC Tower.



(a) NOSC tower wave computer data 2/13/85; reel +3 footage 1190; 17 seconds; maximum crest to trough depth is 1.2 feet; aspect angle is 70 degrees; z axis gain is 8.



(b) NOSC tower wave computer data 2/13/85; reel +3 footage 1190; 34 seconds; maximum crest to trough depth is 0.9 foot; aspect angle is 70 degrees; z axis gain is 8.



(c) NOSC tower wave computer data 2/13/85; reel +3 footage 1190; 60 seconds; maximum crest to trough depth is 1.2 feet; aspect angle is 70 degrees; z axis gain is 8.

FIGURE 12. Reconstructed Sea Surface, 625 Square Feet.

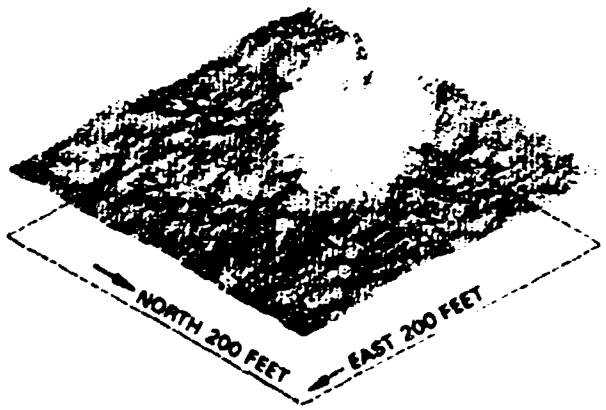
Taking this spatial extension to the limit we find roughly an acre (40,000 square feet) of reconstructed sea surface in Figure 13(a) through (c). Here the wave computer errors become quite eminent. Unlike the foregoing images, there is little correlation between wavelengths and wave directions for various data record lengths. Moreover, the longer wavelength structure is not commensurate with observations recorded at the Tower. This suggests that the wave computer has difficulty in measuring and reproducing longer wavelength information.

There is, however, a quite reasonable explanation for this phenomena. As noted earlier, the first principal frequency component in the "Z" spectrum shown in Table 2 occurs at about 0.059 hertz. By the gravity wave formula this would correspond to a wavelength of 1471 feet long, which, with an amplitude of 1 foot, yields roughly a maximum slope of 0.005. If the wave was as much as 2 feet from the beam screen, this would correspond to a laser beam deflection of 0.24 inch. The wave computer quantizes the position of the laser spot in steps of 0.28 inch, a characteristic of the 8-bit words used in the X and Y vectors. This introduces devastating quantization errors at frequencies below 0.117 hertz. This is evidenced by the measured wavelengths diverging from the theoretical wavelengths shown in Table 2. This evidence is somewhat questionable in that the NOSC Tower is on the edge of the continental shelf in only 50 feet of water, and contraction and distortion of wavelengths from the basin effect is eminent.

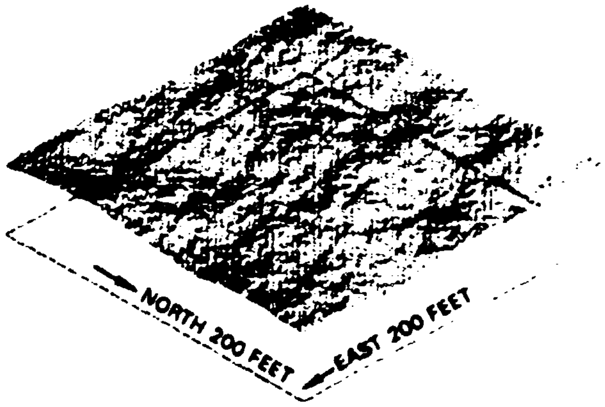
CONCLUDING REMARKS

The wave computer has the potential of providing accurate information on the microstructure of the sea surface. However, its ability to measure and reconstruct long wavelength (low frequency) information is questionable. It would seem simple enough to increase the word size of the X and Y vectors from 8 to 10 bits and modify the beam screen for more precision measurements. This would eliminate problems associated with hardware inadequacies, but it would not address limitations of the data reduction algorithm itself. That is, this entire plane wave superposition model may behave much like a Taylor's Series with its own radius of convergence. Thus, there is a possibility that enhancing the performance of the equipment could result in no additional information.

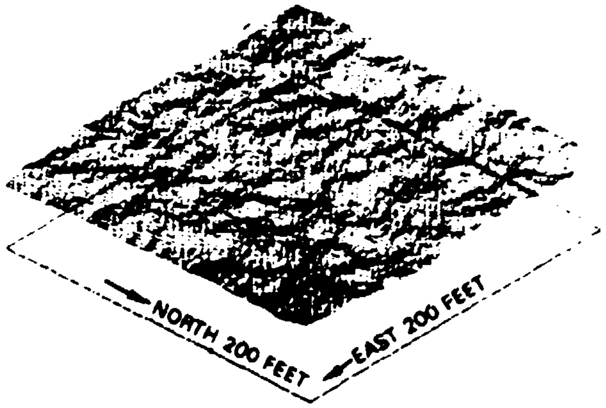
This is not to say that the information obtained is useless. More often than not system designers are concerned with microstructure rather than with long wavelength swells. Moreover, the introduction of swells into the existing mathematical model would be a comparatively simple task. The author does not wish to imply that all the necessary work has



(a) NOSC tower wave computer data 2/13/85; reel +3 footage 1190; 17 seconds; maximum crest to trough depth is 2.4 feet; aspect angle is 70 degrees; z axis gain is 10.



(b) NOSC tower wave computer data 2/13/85; reel +3 footage 1190; 34 seconds; maximum crest to trough depth is 2.7 feet; aspect angle is 70 degrees; z axis gain is 10.



(c) NOSC tower wave computer data 2/13/85; reel +3 footage 1190; 60 seconds; maximum crest to trough depth is 2.5 feet; aspect angle is 70 degrees; z axis gain is 10.

FIGURE 13. Reconstructed Sea Surface, 40,000 Square Feet.

been completed to make this model a reliable standard for a simulation laboratory. There still remains considerable theoretical analysis, which involves treating the wave model parameters as random variables to determine an optimal data record length. Additionally, exhaustive field testing employing parallel redundant imaging of the sea surface is mandatory.

REFERENCES

1. William H. Hayt, Jr. Engineering Electromagnetics. New York, McGraw-Hill, 1974. 140 pp.

2. Harley Flanders, Robert R. Korfhage, and Justin J. Price. Calculus. New York, Academic Press, 1970. 969 pp.

Appendix A

WAVE COMPUTER THEORY OF OPERATION

(1) Sync Strip and Threshold Processing Card

(2) Pulse Generation Logic Card

(3) X Vector Logic Card

(4) Y Vector Logic Card

(5) Blanking Interval Disable Card

(6) Dead Pixel Rejection Card

(7) Capacitive Probe Subsystem

OVERVIEW

The fundamental instrument in this sea-surface measurement effort is the wave computer system. This system consists of a video vectoring system and a parallel capacitive probe network. The video vectoring system is responsbile for determining the position of the Laser spot from the composite video signal of the monitoring video camera. The capacitive probe subsystem is responsible for reporting the vertical elevation of the sea surface at the point of impact of the laser beam.

(1) SYNC STRIP AND THRESHOLD PROCESSING CARD

Figure A-1 is a circuit schematic of the Sync Strip and Threshold Processing Card (Card 1) of the wave computer system. This card is responsible for amplifying the video signal, detecting the occurrence of the laser spot, and stripping off the horizontal and vertical sync pulses for reference purposes. The incoming video signal is buffered by the amplifier shown by label A. From this point the signal is split, and by label F a clamping network facilitates the stripping of the horizontal sync pulses and the integrator shown by label G determines the location of the vertical sync pulses. These signals are used to reference the X and Y vector counters, the dead pixel inhibit counters, and the blanking interval disable circuitry. The upper path of the composite video signal is then fed into an amplifier at label B where its amplitude is boosted about four-fold. At this point the signal is again split with the lower half being used to establish a dynamic threshold at label C. When the composite video signal is on a visible portion of the video line, the network shown by label C performs as a low pass filter with a corner frequency of 3.2 kHz and a gain of −6 dB. However, during the blanking interval of the video signal, the analog switch converts this network into a track-and-hold circuit, thus holding the last analog level experienced before the end of the video line. This allows the threshold level to track changing gray levels in the video background. This threshold level (THL) is then fed into the comparator shown by label E where it is compared to the incoming video signal. A pulse is then output by this comparator when the threshold is crossed, which shall be referred to as threshold crossing (THC).

(2) PULSE GENERATION LOGIC CARD

Turning our attention to Figure A-2 we find the Pulse Generation Logic Card (Card 2). This card is responsible for generating clock pulses for the X and Y vectors to count and to inhibit false threshold crossings. The central wave computer 16-MHz reference clock is shown by label A. The clock is then split and one branch is divided by four as shown by label B to yield the horizontal count pulses (HCP) for the X vector and a threshold window. The threshold window prohibits vector locking during the interval in which the vectors are transient. The other branch of the split signal is then fed into an elaborate divide-by-10,500 counter that outputs a 15,238-hertz clock, which constitutes the vertical count pulses (VCP) for use by the Y vector. Referring now above label E, the incoming threshold crossing signal (THC) is gated by the threshold inhibit signal (THIN) that precludes the locking of data vectors in an area known to be in a blanking interval or dead pixel location. When a valid threshold crossing is experienced, the D-type flip-flop is set, which locks the X and Y data vectors in the adjacent cards. At the same time, a one-shot is fired outputting a 25-microsecond interrupt pulse informing external devices that data has been acquired. Labels F and G indicate buffers and line drives used to strengthen these signals for external transport.

(3) X VECTOR LOGIC CARD

Figure A-3 is the X Vector Logic Card (Card 3). This card is responsible for locking in the correct X or horizontal coordinate of the laser spot when a valid threshold crossing has been detected. Referring now to label A we find two cascaded 4-bit counters that count the horizontal count pulses (HCPB) and are reset at the beginning of each video line by the horizontal sync pulse (HSB).* At label B we find two 4-bit latches that lock in the current value of these counters when a valid threshold crossing occurs. The output of these latches is then fed into a D/A converter shown by label C, which converts this digital X vector into an analog signal that is denoted as x(t). This analog signal is then split, with one branch going to the panel meter for calibration purposes and the other being output to a BNC jack for recording on the instrumentation tape recorder. This card ideally updates its output 60 times a second; however, if no valid threshold crossing occurs, this card will hold onto the last X coordinate it observed.

(4) Y VECTOR LOGIC CARD

Referring to Figure A-4 we find the Y Vector Logic Card (Card 4). This card operates in the exact same fashion as the X Vector Logic Card only with different inputs. Instead of counting horizontal count pulses, the Y Vector Card counts vertical count pulses (VCPB). Also, rather than the counters being reset at the beginning of each video line, the counters are reset at the beginning of each video field by the vertical sync pulse (VSB). Other than these differences, the operation of these cards is identical.

* In this appendix, "B" in an abbreviation indicates buffered.

(5) BLANKING INTERVAL DISABLE CARD

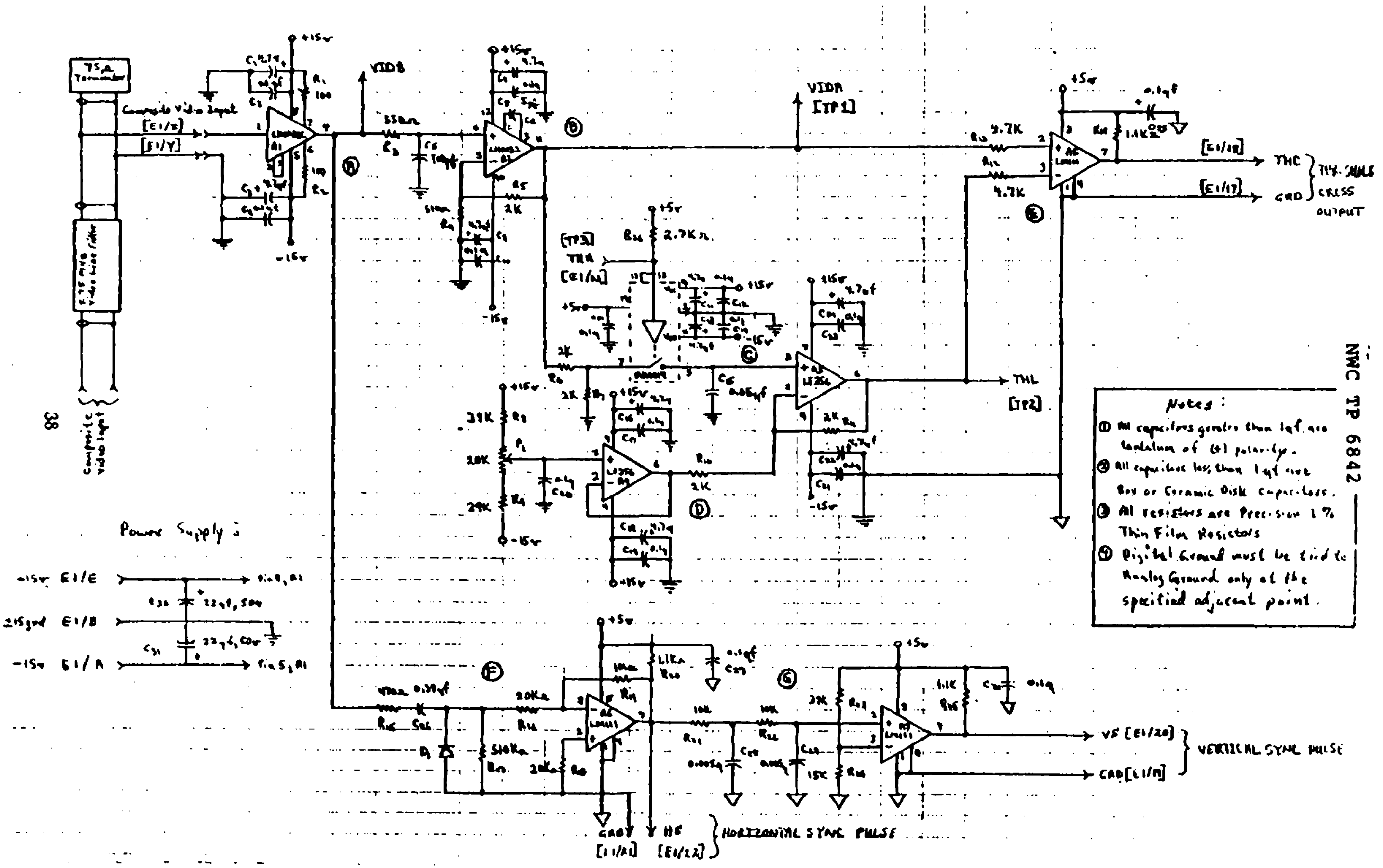
Figure A-5 is the Blanking Interval Disable Card (Card 5). This card is responsible for outputting a pulse when the monitoring video camera is in a blanking interval or scanning over a dead pixel. The card is divided into two independent horizontal and vertical channels each of which is responsible for outputting its corresponding blanking signal. Label A shows the horizontal sync pulse (HSB), which sets the flip-flop that enables the counters shown by label C to start counting horizontal count pulses (HCPB) at the beginning of the current video line. When the counter time runs out, the carry bit is set, which in turn fires a one-shot that resets the counter enable flip-flop and also fires an adjacent one-shot. The latter one-shot outputs a pulse that is denoted as the horizontal blanking disable pulse (HBD). The vertical blanking disable pulse (VBD) is generated in much the same manner using only different input signals. These two blanking interval disable signals are then ORed together to yield the threshold abort signal (THA), which controls switching of the dynamic threshold of Card 1. Lastly, dead-pixel coordinates are entered from Card 6 and ORed with the blanking interval signals, which then provide the threshold inhibit signal (THIN). The THIN signal is used to inhibit false threshold crossings on Card 2.

(6) DEAD PIXEL REJECTION CARD

Turning our attention to Figure A-6 we find the Dead Pixel Rejection Card (Card 6). This card is responsible for finding and indicating the video scanning of a dead pixel. The detection phase of operation is initialized by the depression of the Reset Switch shown by label G. When this mode is invoked, the RAM chip shown by label E is placed in the write mode with the address lines tied to a surrogate Y vector counter and the data lines are tied to logical highs. Thus, the hexadecimal quantity $FF is written into the first page (256 words) of the RAM chip in one video frame. The dead pixel locations are then mapped by depressing the Search Switch also shown by label G. In the search mode, all threshold crossings in a dark background are assumed to be dead pixels. The surrogate X and Y vectors shown by labels B and A are locked into the 8-bit latches shown by the labels D and C, respectively, when a threshold signal is sensed. During the next vertical blanking interval, the coordinates of the first encountered dead pixel are written into memory with the Y vector constituting the address and the X vector constituting the data element. Thus, this system permits the mapping of only one dead pixel per video line. Once the dead pixels have been mapped, the card defaults to the inhibit mode. In this mode of operation, the Y vector continuously sweeps the address lines of the RAM chip that outputs the location of the dead pixel for that line. When the X vector counts up to the X coordinate output by the RAM data lines, the digital comparators shown by label F output a pulse that is referred to as the dead pixel inhibit signal (DPIN). The DPIN signal is then ORed with blanking interval disable signals to generate the threshold inhibit signal (THIN), which is used to disable false threshold crossings on Card 1. It should be noted that if no dead pixels occur on a specific video line, the RAM chip outputs a hexadecimal $FF that corresponds to somewhere in the horizontal blanking interval, and thus no comparator output is possible.

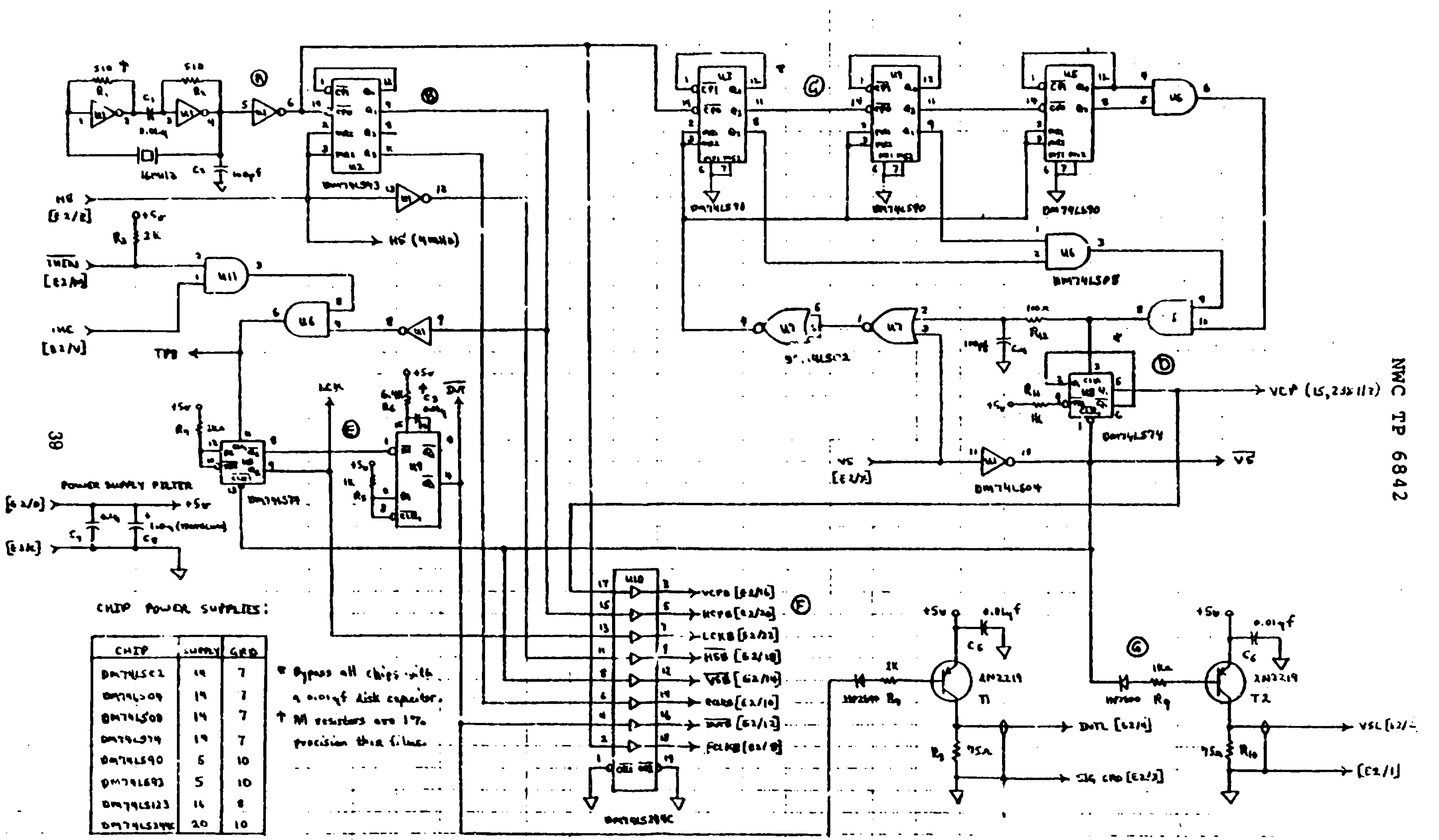
(7) CAPACITIVE PROBE SUBSYSTEM

The Capacitive Probe Subsystem is responsible for reporting the elevation of the sea surface at the point of contact. Although five probes were initially deployed, that is, in the center and on each corner of the beam screen, only the center probe was used in the data reduction phase. Referring now to Figure A-7(a) we find a circuit schematic of a single channel of the Capacitive Probe Subsystem. Focusing our attention on label A we find the 1-MHz central capacitive probe reference clock. This transistor-transistor-logic level (TTL) clock is then divided down to 10 kHz by the counters shown by label B. The 10-kHz clock is then boosted to a +15-volt-CMOS logic level by the comparator shown by label C. At label D we find the core of the system, a CMOS monostable one-shot whose output pulse duration is dependent on the capacitance across its terminals labeled pin numbers 1 and 2. The capacitive probes themselves constitute coaxial cables using the sea surface for an outer conductor. Thus, the capacitance perceived at the terminals is dependent solely on the elevation of the sea surface, which determines the duration of the output pulse. It should be noted that the transistor diode network shown by label E is used to squelch the residual energy trapped in the LC tank circuit indigenous to the capacitive probe. This is done by shorting the probe to ground during the first 5 microseconds of the dead cycle. Referring now to label F we find a four-pole 160-hertz low-pass filter that performs the role of an integrator of these one-shot output pulses. The result at label G is then a buffered DC level directly proportional to the elevation of the sea surface. This signal is then transported to a simple offset and scaling operational amplifier (Figure A-7(d)) for calibration before it is recorded on the instrumentation recorder. In Figures A-7(b) and A-7(c) the full five-channel capacitive probe circuit schematic is shown.



Notes:
1. All capacitors greater than 1μf are tantalum of (+) polarity.
2. All capacitors less than 1μf are Box or Ceramic Disk capacitors.
3. All resistors are Precision 1% Thin Film Resistors
4. Digital Ground must be tied to Analog Ground only at the specified adjacent point.

FIGURE A-1. Sync Strip and Threshold Processing Card (Card 1).



CHIP POWER SUPPLIES:

CHIP	SUPPLY	GRD
DM74LS02	14	7
DM74LS04	14	7
DM74LS08	14	7
DM74LS74	14	7
DM74LS90	5	10
DM74LS93	5	10
DM74LS123	16	8
DM74LS244C	20	10

* Bypass all chips with a 0.01μf disk capacitor.

† All resistors are 1% precision thin films.

FIGURE A-2. Pulse Generation Logic Card (Card 2).

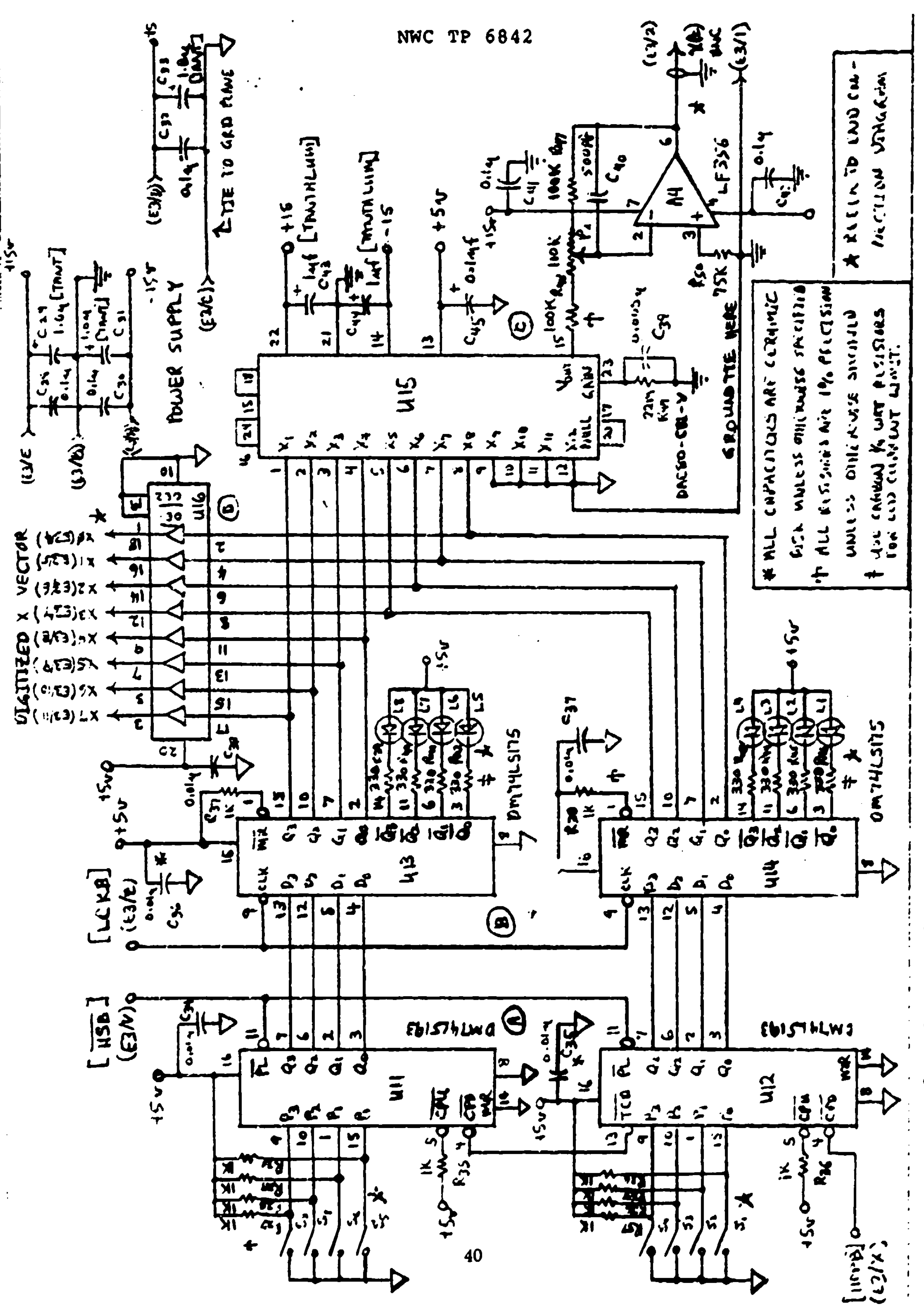

DIGITIZED X VECTOR
POWER SUPPLY
TIE TO GRD PLANE
GROUND TIE HERE
U11
U12
U13
U14
U15
U16
DM74LS193
DM74LS175
DAC80-CBI-V
LF356
A1
[LCKB]
[HSB]
* ALL CAPACITORS ARE CERAMIC DISK UNLESS OTHERWISE SPECIFIED
† ALL RESISTORS ARE 1% PRECISION UNLESS OTHERWISE SPECIFIED
‡ USE CARBON ¼ WATT RESISTORS FOR LED CURRENT LIMIT.
★ REFER TO LAND CONNECTION DIAGRAM

FIGURE A-3. X Vector Logic (Card 3).

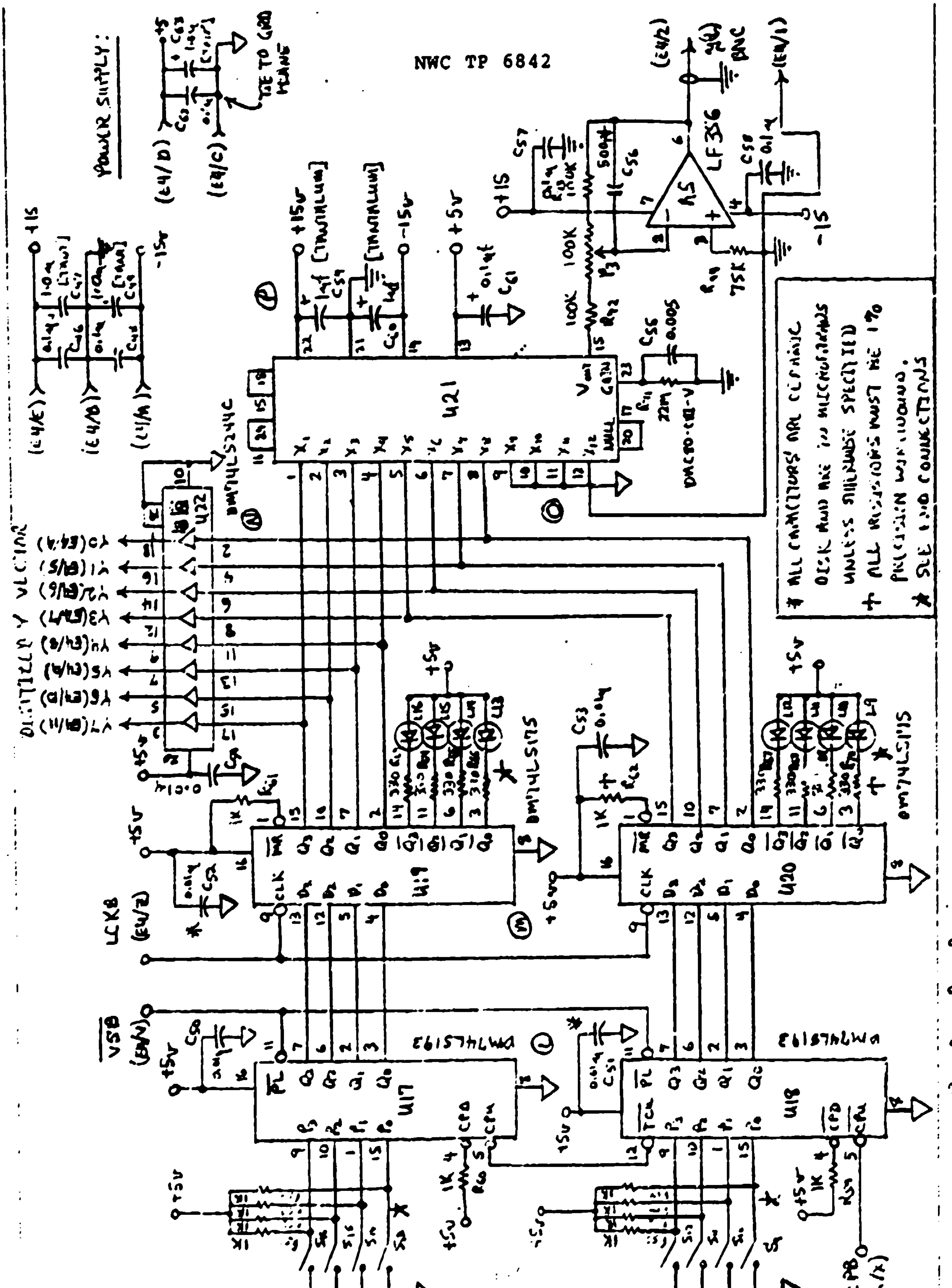


FIGURE A-4. Y Vector Logic (Card 4).

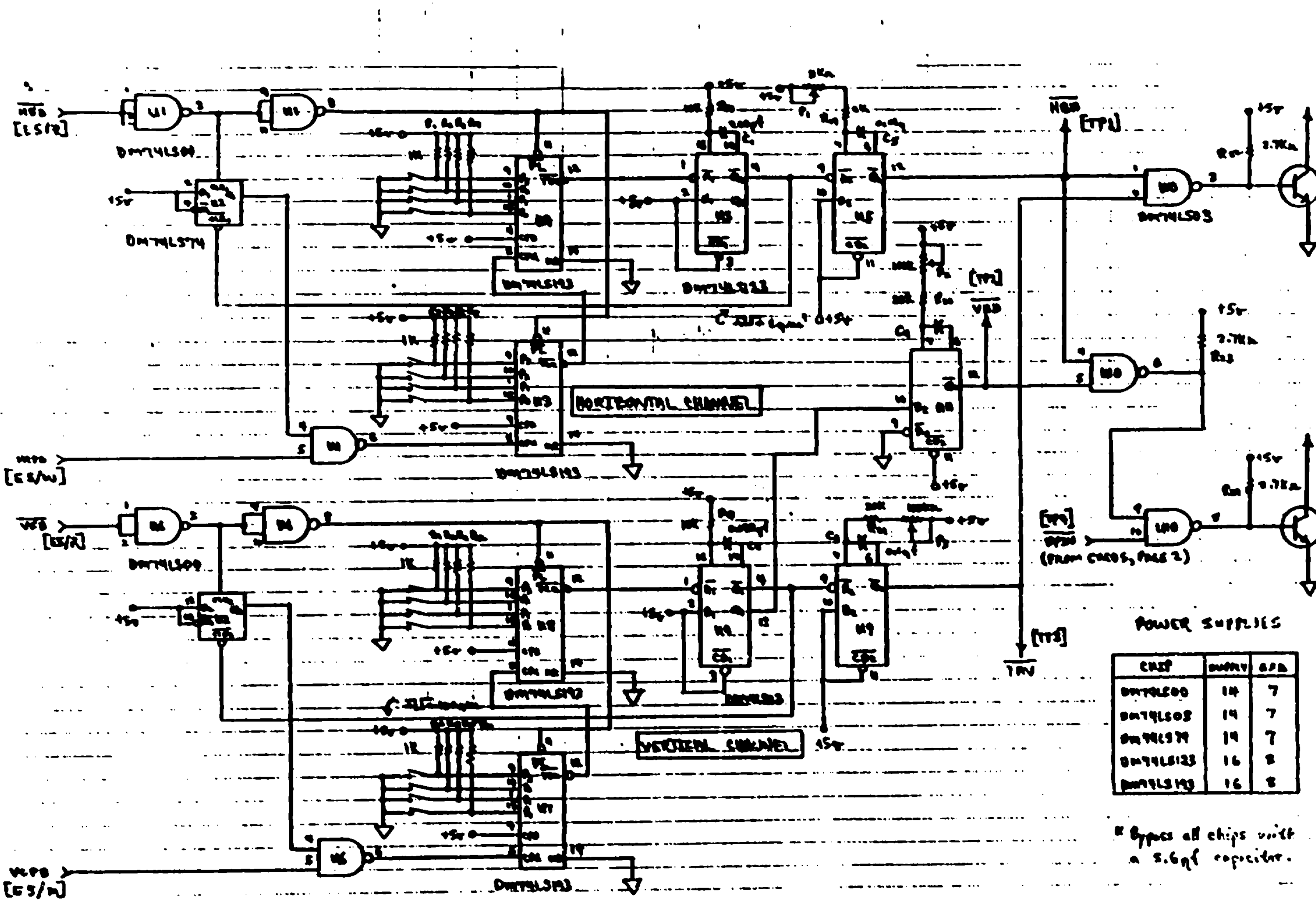
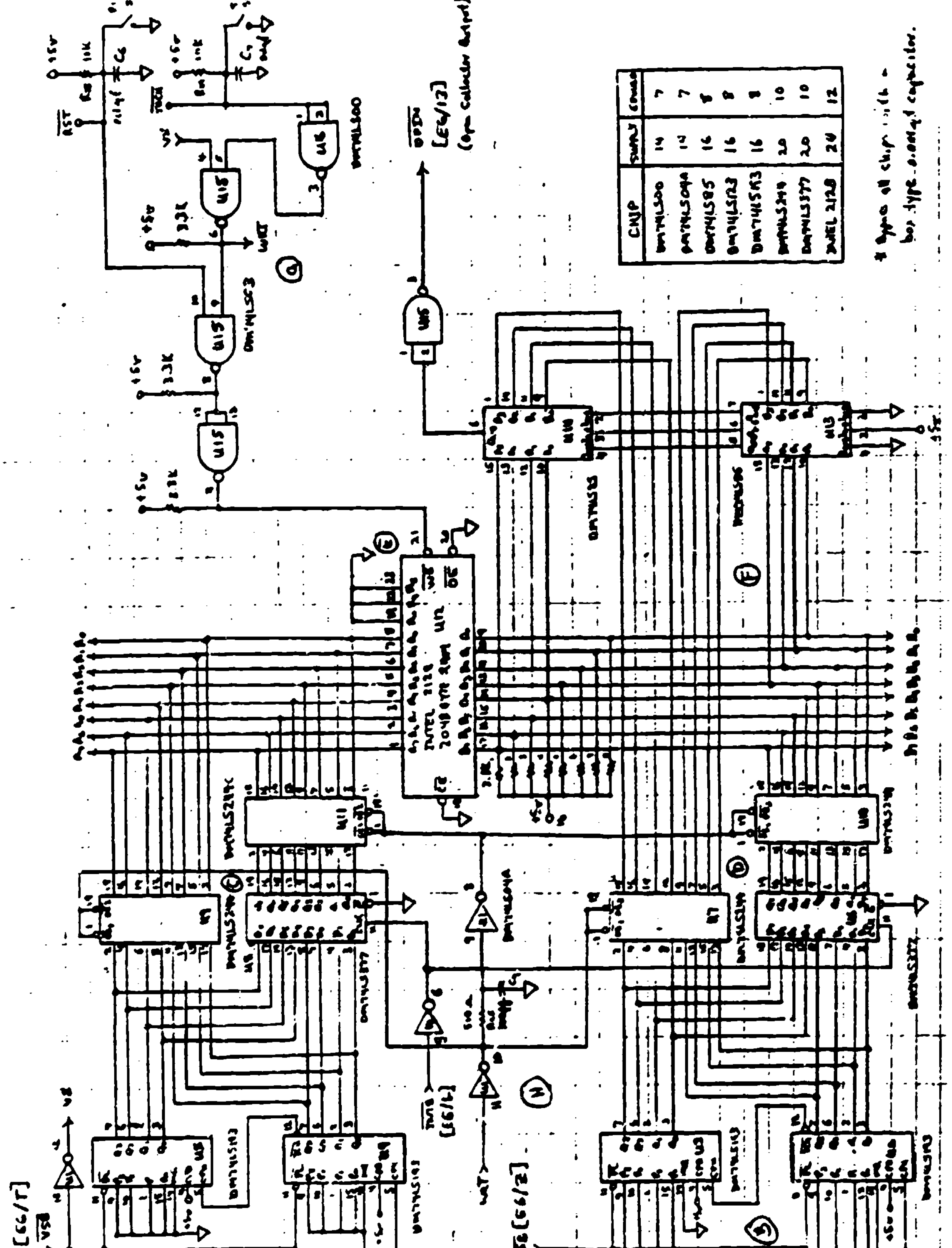


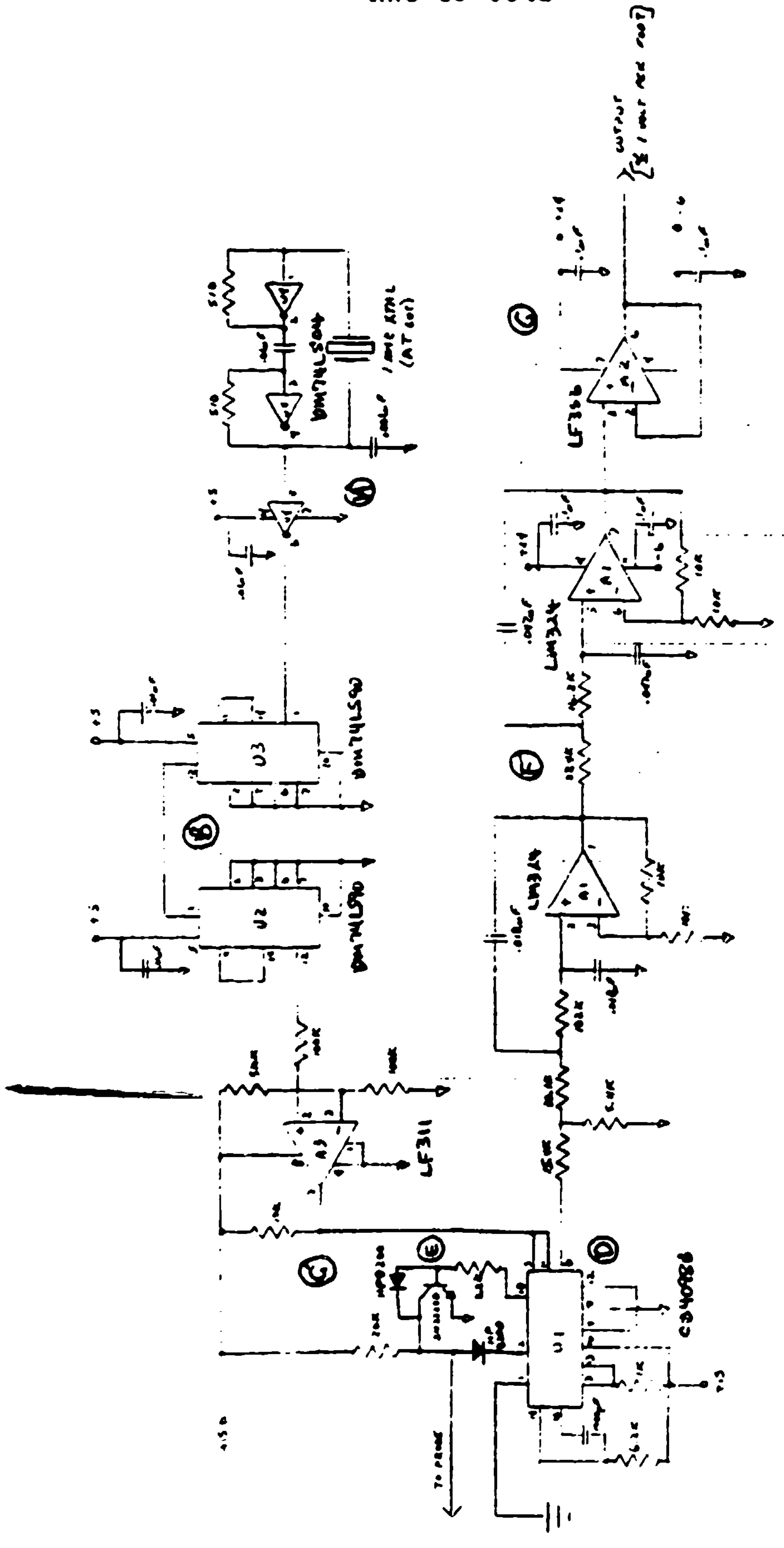
FIGURE A-5. Video Signal Discrimination Card (Blanking Interval Disable Card, Card 5).



CHIP	SUPPLY	GROUND
DM74LS00	14	7
DM74LS04A	14	7
DM74LS85	16	8
DM74LS132	16	8
DM74LS163	16	8
DM74LS240	20	10
DM74LS377	20	10
INTEL 2128	24	12

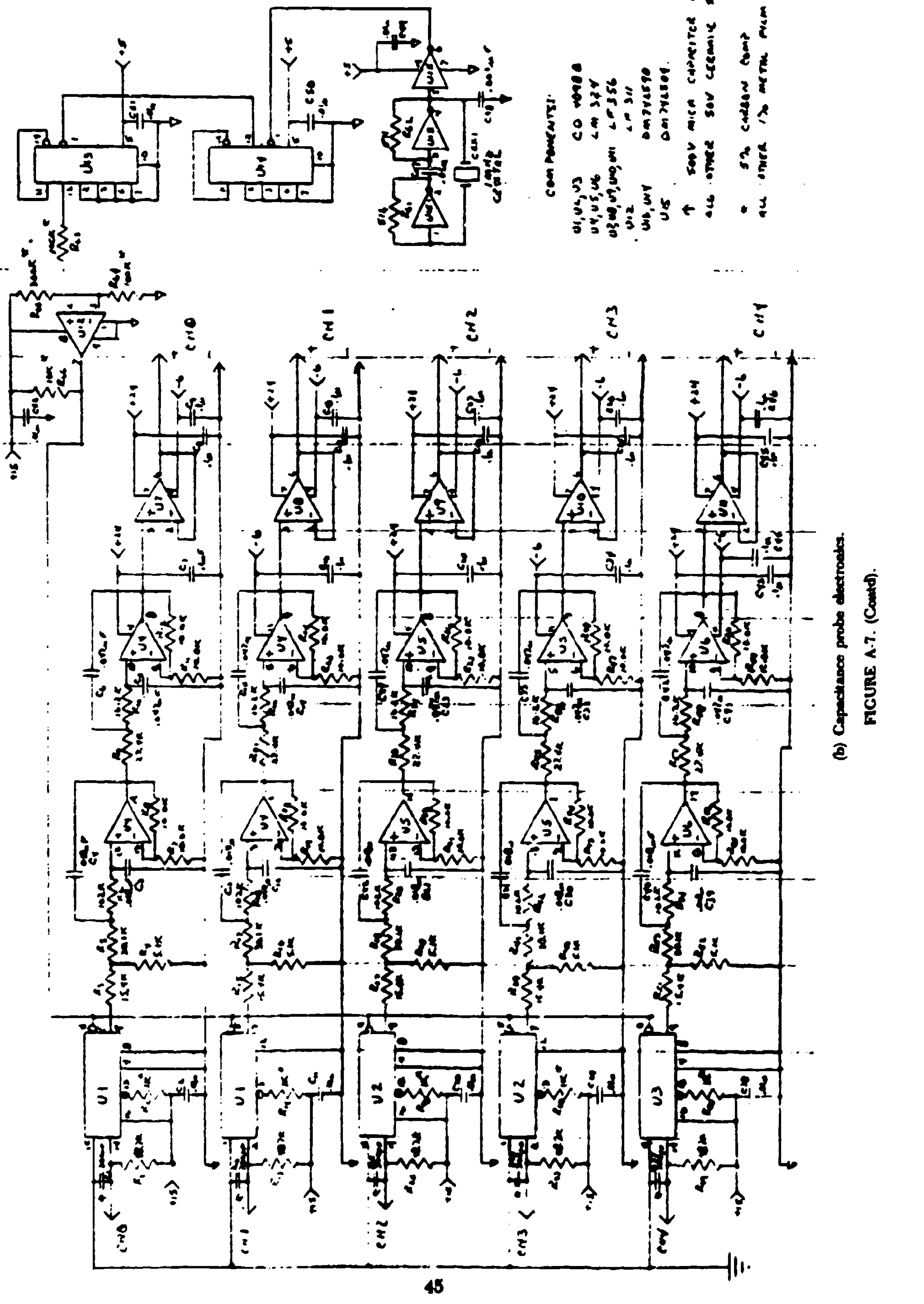
* Bypass all chips with a bar type 0.047 µf capacitor.

FIGURE A-6. Video Signal Discrimination Card (Dead Pixel Rejection Card, Card 6).



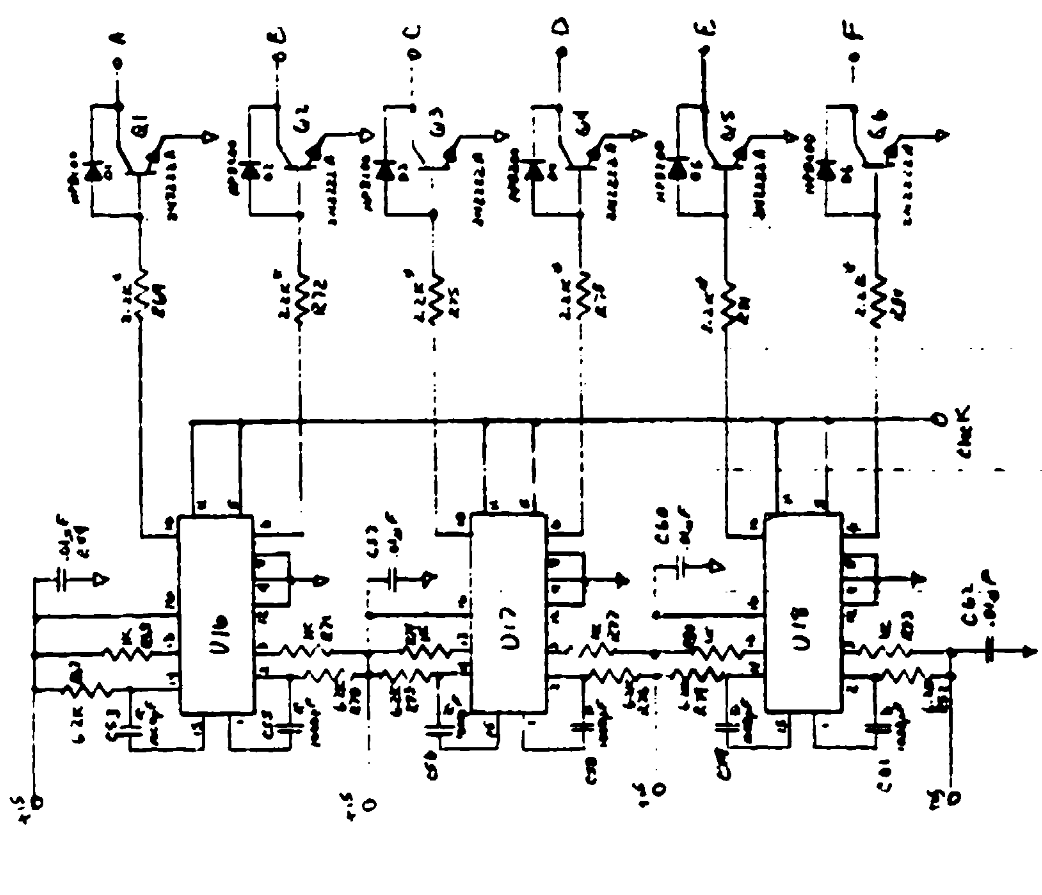
(a) Single channel of capacitive probe signal conditioner.

FIGURE A-7. Capacitive Probe Subsystem Schematic.



(b) Capacitance probe electronics.

FIGURE A-7. (Contd).



COMPONENTS:

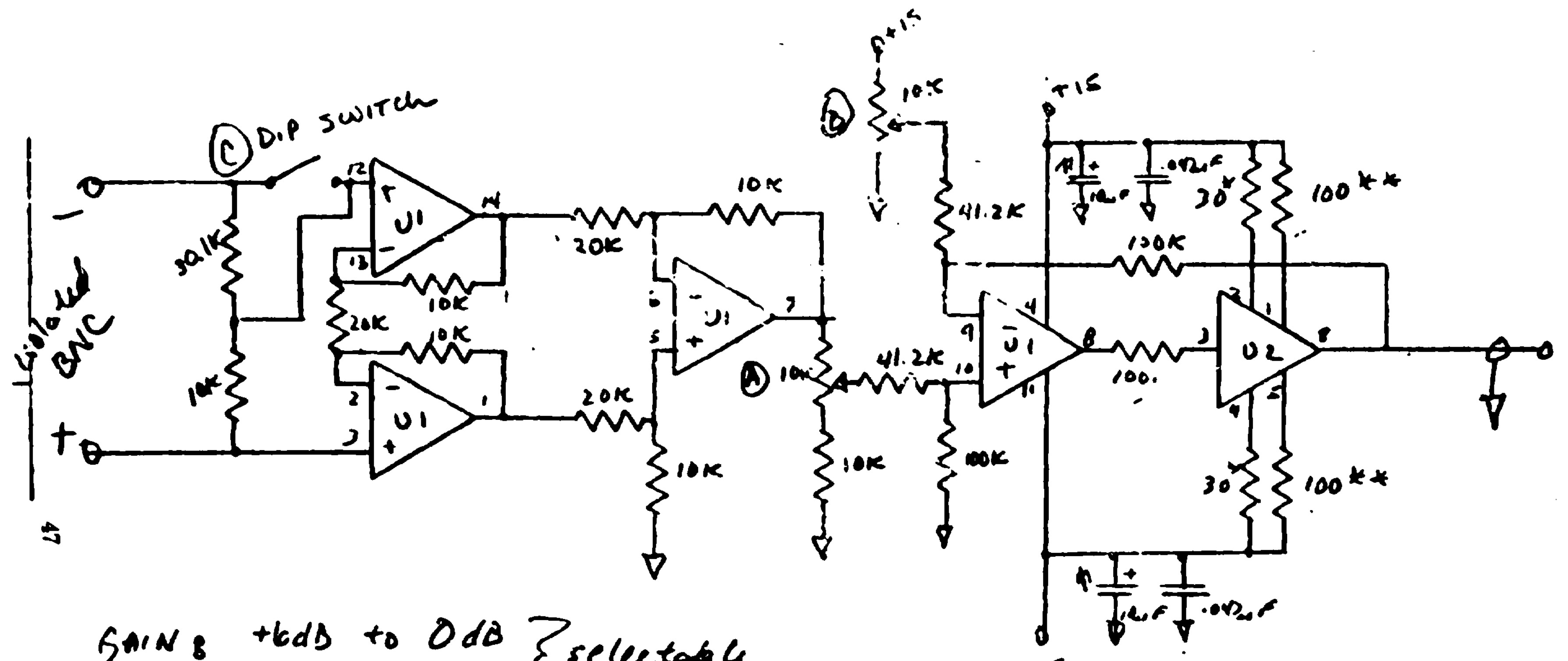
U16, U17, U18 CD4028B

¶ 500V MICA 1% CAP.
ALL OTHER CAP - CERAMIC 5%

* CARBON COMP. 5% RESISTOR
ALL OTHER RESISTOR 1% METAL FILM

(c) Capacitance probe electronics (modification board).

FIGURE A-7. (Contd).



GAIN: +6dB to 0dB } selectable.
-6dB to -12dB }

OFFSET: 0 to -15 VDC

Ⓐ GAIN ADJUST
Ⓑ OFFSET
Ⓒ GAIN RANGE SELECT

U1	LM324
U2	LH0002
*	$\frac{1}{4}$ω carbon comp
**	$\frac{1}{2}$ω " "

ALL OTHER 1/4 ω Metal Film
All ALUMINUM ELECTROLYTIC

(d) Beam screen instrumentation amplifier / driver, offset and scaling circuitry.

FIGURE A-7. (Contd).

Appendix B

FOURIER THEORETICAL PROOFS

(1) Fourier Transform of Sea Surface Elevation

(2) Fourier Transform of Spatial Partial Derivatives

(1) FOURIER TRANSFORM OF SEA SURFACE ELEVATION

In this section of Appendix B we will derive the Fourier transform of the sea surface elevation. All variables used here are defined in the body of the report. We begin this derivation by direct application of the Fourier transform to our sea surface model. Thus,

$$F\{Z_o(t)\} = \int_{-\infty}^{\infty} Z_o(t)e^{-j\omega t}dt =$$

$$\int_{-P/2}^{P/2} \left\{ \sum_{n=0}^{N-1} C_n \cos(\omega_n t - \psi_n) \right\} e^{-j\omega t}dt \qquad \text{(B-1)}$$

Expanding the cosine term we obtain,

$$F\{Z_o(t)\} = \sum_{n=0}^{N-1} C_n \left[\cos(\psi_n) \int_{-P/2}^{P/2} \cos(\omega_n t)e^{-j\omega t}dt \right.$$

$$\left. + \sin(\psi_n) \int_{-P/2}^{P/2} \sin(\omega_n t)e^{-j\omega t}dt \right] \qquad \text{(B-2)}$$

Defining the integrals,

$$I_1(\omega) = \int_{-P/2}^{P/2} \cos(\omega_n t)e^{-j\omega t}dt \qquad \text{(B-3)}$$

and

$$I_2(\omega) = \int_{-P/2}^{P/2} \sin(\omega_n t)e^{-j\omega t}dt \qquad \text{(B-4)}$$

We may now write,

$$F\{Z_o(t)\} = \sum_{n=0}^{N-1} C_n\left[\cos(\psi_n)I_1(\omega) + \sin(\psi_n)I_2(\omega)\right] \tag{B-5}$$

Now consider the integral,

$$I_1(\omega) = \int_{-P/2}^{P/2} \cos(\omega_n t)e^{-j\omega t}dt = \frac{1}{2}\int_{-P/2}^{P/2}\left[e^{j\omega_n t} + e^{-j\omega_n t}\right]e^{-j\omega t}dt \quad \text{(B-6)}$$

$$I_1(\omega) = \frac{1}{2}\int_{-P/2}^{P/2}\left[e^{j[\omega_n-\omega]t} + e^{-j\omega[\omega_n+\omega]t}\right]dt = \frac{1}{2j}\left\{\frac{e^{j[\omega_n-\omega]t}}{[\omega_n-\omega]} - \frac{e^{-j[\omega_n+\omega]t}}{[\omega_n+\omega]}\right\}\Bigg|_{-P/2}^{P/2} \quad \text{(B-7}$$

Now with $\omega = 2\pi f$ we may further write,

$$I_1(f) = \left\{\frac{\sin[\pi P(f_n-f)]}{2\pi[f_n-f]} + \frac{\sin[\pi P(f_n+f)]}{2\pi[f_n-f]}\right\} \tag{B-8}$$

By exploiting the definition of the sinc function, that is,

$$\text{sinc}(Z) = \frac{\sin(\pi Z)}{\pi Z} \tag{B-9}$$

we may now write Equation B-8 as,

$$I_1(f) = \frac{P}{2}\,\text{sinc}[P(f_n - f)] + \frac{P}{2}\,\text{sinc}[P(f_n + f)] \tag{B-10}$$

In a similar manner it is easily shown that the second defined integral results in,

$$I_2(\omega) = \int_{-P/2}^{P/2} \sin(\omega_n t)e^{-j\omega t}dt = \frac{1}{2j}\int_{-P/2}^{P/2}\left[e^{j[\omega_n-\omega]t} - e^{-j[\omega_n+\omega]t}\right]dt \quad \text{(B-11)}$$

$$I_2(f) = \frac{P}{2j}\operatorname{sinc}[P(f_n - f)] - \frac{P}{2j}\operatorname{sinc}[P(f_n + f)] \quad \text{(B-12)}$$

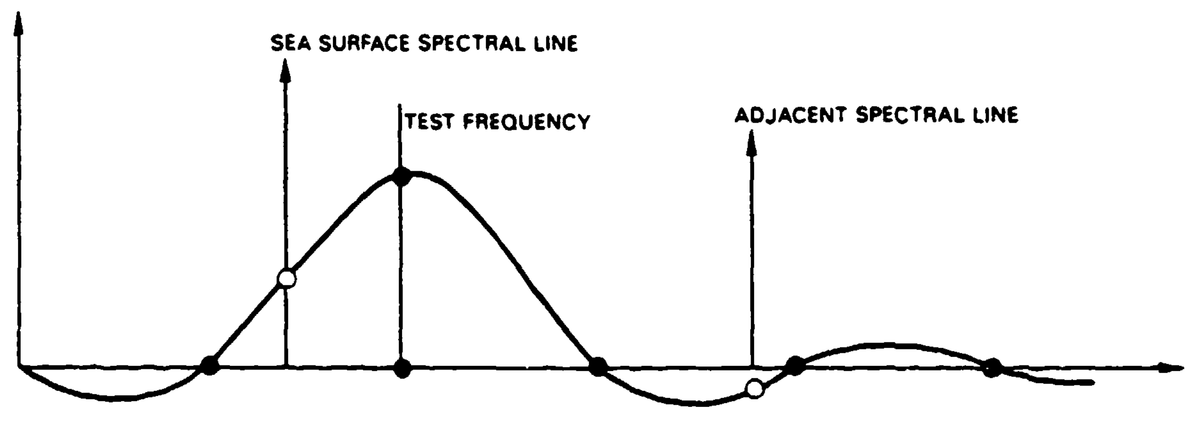
If we consider only the sinc terms of Equations B-10 and B-12 that peak for positive frequencies and substitute them into Equation B-5 we obtain,

$$F\{Z_o(t)\} = \sum_{n=0}^{N-1} C_n \left\{\frac{P}{2}\operatorname{sinc}[P(f_n - f)]\cos(\psi_n) \quad 2j\right.$$

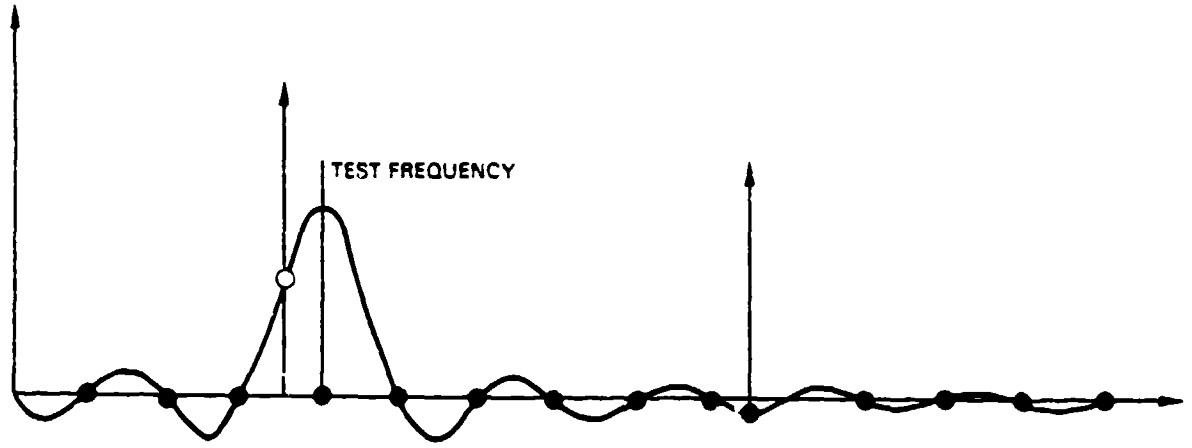
$$\left. + \frac{P}{2j}\operatorname{sinc}[P(f_n - f)]\sin(\psi_n)\right\} \quad \text{(B-13)}$$

$$F\{Z_o(t)\} = \sum_{n=0}^{N-1}\left[\frac{PC_n}{2}\right]\operatorname{sinc}[P(f_n - f)]\left\{\cos(\psi_n) - j\sin(\psi_n)\right\} \quad \text{(B-14)}$$

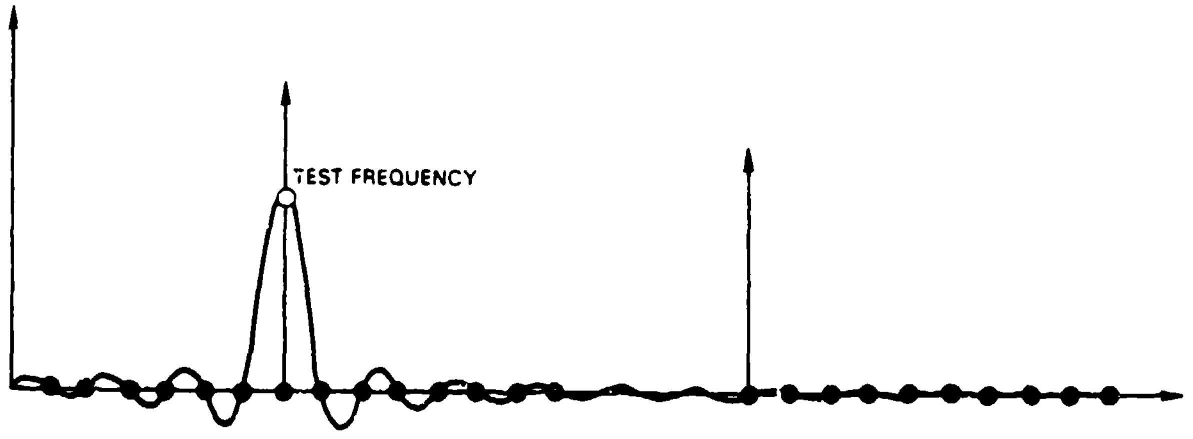
At this point some critical assumptions must be made to isolate spectral components. Consider the sinc pulse shown in Figure B-1(a). The spectral line of a wave component (shown as a vertical arrow) may be anywhere on the frequency axis. The dots on the frequency axis would be points tested by our discrete Fourier transform. Note that the point tested in Figure B-1(a) may also include the sinc weighted contribution of another spectral line. However, if one considers protracting the period of observation, we obtain much less interference from adjacent spectral lines as shown in Figure B-1(b). If we were to extend this period of observation "P" to an optimal length, all terms except for the tested point would tend toward zero, as shown in Figure B-1(c).



(a) P_o second observation.



(b) $\frac{5}{2}$ P_o second observation.



(c) 5 P_o second observation.

FIGURE B-1. Sea Surface Frequency Spectrum.

With the assumption of an optimal observation period, and evaluating Equation B-14 at some arbitrary frequency step, say $f = f_k$, all terms in the series with the exception of the tested point would tend toward zero, leaving us with,

$$F\{Z_o(t)\}\Big|_{f_k} = \left[\frac{PC_k}{2}\right] \operatorname{sinc}[P(f_n - f_k)]\ \{\cos(\psi_k) - j\sin(\psi_k)\} \qquad \text{(B-15)}$$

By taking the magnitude of Equation B-15 and assuming that the sinc term is very close to unity we obtain,

$$\left|F\{Z_o(t)\}\right|\Big|_{f_k} = \frac{PC_k}{2}\left|\cos(\psi_k) - j\sin(\psi_k)\right| = \frac{PC_k}{2} \qquad \text{(B-16)}$$

The last step of this process entails the proper weighting of this spectral component. That is, we are not taking a true Fourier integral transform of this data. Instead, we are approximating this operation with a discrete Fast Fourier Transform (FFT). Ideally, with all quantization errors aside, this should differ from the Fourier integral transform by the quantized temporal differential, that is,

$$|\hat{Z}_o(f_k)| = \frac{1}{T_s}\left|F\{Zo(t)\}\right|\Big|_{f_k} = \frac{PC_k/2}{P/N_p} = \frac{N_p}{2}C_k \qquad \text{(B-17)}$$

where

N_p = number of points in the FFT

Equation B-17 is the final result referenced in the body of the report.

(2) FOURIER TRANSFORM OF SPATIAL PARTIAL DERIVATIVES

In this section we will be taking the Fourier transform of the first spatial partial derivative of the sea surface elevation with respect to "x." We will then extend the results to determine the Fourier transform of the first spatial partial derivative of the sea surface with respect to "y." We begin the derivation by differentiating the sea surface model with respect to "x."

$$Z(\vec{r},t) = \sum_{n=0}^{N-1} C_n \cos\left[\frac{2\pi}{\lambda_n}(x\cos\gamma_n + y\sin\gamma_n - V_n t) - \psi_n\right] \qquad \text{(B-18)}$$

Differentiating with respect to "x" and evaluating at origin we obtain,

$$\frac{\partial Z(\vec{o},t)}{\partial x} = \frac{\partial Zo(t)}{\partial x} = -2\pi \sum_{n=0}^{N-1} \left(\frac{C_n}{\lambda_n}\right) \cos(\gamma_n)\ \sin(-\omega_n t - \psi_n)^{*} \qquad \text{(B-19)}$$

Expanding the temporal sinusoid we obtain,

$$\frac{\partial Z_o(t)}{\partial x} = 2\pi \sum_{n=0}^{N-1} \left(\frac{C_n}{\lambda_n}\right) \cos(\gamma_n) \left\{\sin(\omega_n t)\cos(\psi_n) + \cos(\omega_n t)\sin(\psi_n)\right\} \quad \text{(B-20)}$$

Applying the Fourier transform to Equation B-20 then results in,

$$F\left\{\frac{\partial}{\partial x} Z_o(t)\right\} = \int_{-\infty}^{\infty} 2\pi \sum_{n-0}^{N-1} \left(\frac{C_n}{\lambda_n}\right) \cos(\gamma_n) \Big[\sin(\omega_n t)\cos(\psi_n) + \cos(\omega_n t)\sin(\psi_n)\Big] e^{-j\omega t} dt \qquad \text{(B-21)}$$

Interchanging the order of integration and summation and temporally windowing our transform, we obtain,

$$F\left\{\frac{\partial}{\partial x} Z_o(t)\right\} = 2\pi \sum_{n=0}^{N-1} \left(\frac{C_n}{\lambda_n}\right) \cos(\gamma_n) \left\{\cos(\psi_n) \int_{-P/2}^{P/2} \sin(\omega_n t) e^{-j\omega t} dt + \sin(\psi_n) \int_{-P/2}^{P/2} \cos(\omega_n t) e^{-j\omega t} dt\right\} \qquad \text{(B-22)}$$

---

* Note: $\omega_n = \frac{2\pi V_n}{\lambda_n}$

$$F\left\{\frac{\partial Z_o(t)}{\partial x}\right\} = 2\pi\sum_{n=0}^{N-1}\left(\frac{C_n}{\lambda_n}\right)\cos(\gamma_n)\left\{\cos(\psi_n)I_2(\omega) + \sin(\psi_n)I_1(\omega)\right\} \qquad \text{(B-23)}$$

Fortunately, we have already determined the integrals $I_1(\omega)$ and $I_2(\omega)$ earlier in this appendix (Equations B-10 and B-12, respectively). Again by considering only positive frequency sensitive sine functions we may write Equation B-23 as,

$$F\left\{\frac{\partial Z_o(t)}{\partial x}\right\} = \pi\sum_{n=0}^{N-1}\left(\frac{PC_n}{\lambda_n}\right)\cos(\gamma_n)\ \mathrm{sinc}[P(f_n - f)]\left\{\sin(\psi_n) - j\cos(\psi_n)\right\} \qquad \text{(B-24)}$$

On the basis of the earlier optimal observation period argument, we may evaluate Equation B-24 at a specific frequency, f_k, and extract its magnitude yielding,

$$\left|F\left\{\frac{\partial Z_o(t)}{\partial x}\right\}\bigg|_{f_k}\right| = \left[\frac{\pi PC_k}{\lambda_k}\right]|\cos(\gamma_k)| \qquad \text{(B-25)}$$

Again, we must scale this Fourier integral transform by the quantized differential so that our results will be commensurate with our FFT operation. Thus,

$$\boxed{|\hat{Z}_x(f_k)| \quad \frac{N_p}{P}\left|F\left\{\frac{\partial Z_o(t)}{\partial x}\right\}\bigg|_{f_k}\right| = \left[\frac{\pi N_p C_k}{\lambda_k}\right]|\cos(\gamma_k)|} \qquad \text{(B-26)}$$

Equation B-26 is the final form of the spectral component referred to in the body of the report.

We are now in a position to extend the results of the Fourier transform to $\partial Z_o(t)/\partial x$ to $\partial Z_o(t)/\partial y$. Again, we will begin by differentiating the sea surface model.

$$Z(r,t) = \sum_{n=0}^{N-1} C_n \cos\left[\frac{2\pi}{\lambda_n}(x\cos\gamma_n + y\sin\gamma_n - V_n t) - \psi_n\right] \quad \text{(B-27)}$$

Differentiating with respect to "y" and evaluating at origin yields,

$$\frac{\partial Z(\vec{o},t)}{\partial y} = \frac{\partial Z_o(t)}{\partial y} = -2\pi \sum_{n=0}^{N-1} \left(\frac{C_n}{\lambda_n}\right) \sin(\gamma_n) \sin\left(\frac{-2\pi V_n t}{\lambda_n} - \psi_n\right) \quad \text{(B-28)}$$

$$\frac{\partial Z_o(t)}{\partial y} = 2\pi \sum_{n=0}^{N-1} \left(\frac{C_n}{\lambda_n}\right) \sin(\gamma_n) \sin(\omega_n t + \psi_n) \quad \text{(B-29)}$$

$$\frac{\partial Z_o(t)}{\partial y} = 2\pi \sum_{n=0}^{N-1} \left(\frac{C_n}{\lambda_n}\right) \sin(\gamma_n) \left[\sin(\omega_n t)\cos(\psi_n) + \cos(\omega_n t)\sin(\psi_n)\right] \quad \text{(B-30)}$$

Comparing Equation B-30 with Equation B-20 it becomes clear that the two differ only by a $\sin(\gamma_n)$ in place of a $\cos(\gamma_n)$. Since this sole difference in form is but a constant that may be pulled out of the Fourier transform operation, we may conclude that,

$$\left|\hat{Z}y(f_k)\right| = \left[\frac{\pi N_p C_k}{\lambda_k}\right] \left|\sin(\gamma_k)\right| \quad \text{(B-31)}$$

Equation B-31 is the final form referred to in the body of the report.

Appendix C

FIELD TEST DATA INTERFACING SOFTWARE

31 Aug 1987   20:21:55

```
1000 !**************************************************************
1010 !*********************** PROGRAM SEA_LINK ***********************
1020 !**************************************************************
1030 !*          THIS PROGRAM  IS  RESP[illegible]IBLE  FOR CONTROLLING THE HP6942A *
1040 !* MULTIPROGRAMMER  S[illegible]H  THA[illegible] COMPUTER DATA FROM THE  HONEYWELL *
1050 !* INSTRUMENTATION  T[illegible] RECO[illegible]S CORRECTLY DIGITIZED , SCALED AND *
1060 !* STORED ON DISK FOR LATER R[illegible]VAL .                                 *
1070 !**************************************************************
1080 DIM T(5000),X(5000),Y(5000),Z(5000),D(5000),Cards$[12],Penc(4)
1090 COM /Read_memory/ X_scale,Y_scale,Z_scale
1100 DIM Name$[16],Job$[80]
1110 !****************************************************
1120 !************** DEFINITION OF PROGRAM VARIABLES *************
1130 !****************************************************
1140 Cards$="1,3,5,6,7,8" ! DEFINE ACTIVE MULTIPROGRAMMER CARDS.
1150 Hpib=323             ! MAIN HPIB MULTIPROGRAMMER ADDRESS.
1160 Hpib_mi=32305        ! DEFINE 'MEMORY INPUT' SUB-ADDRESS.
1170 Hpib_int=32309       ! INTERRUPT MULTIPROGRAMMER SUB-ADDRESS.
1180 F_sample=60          ! DEFINE MEAN SAMPLE RATE OF PROCESS.
1190 X_scale=1.5913/200   ! DEFINE 'X' VECTOR SCALE FACTOR .
1200 Y_scale=2.1213/200   ! DEFINE 'Y' VECTOR SCALE FACTOR .
1210 Z_scale=5.656/2000   ! DEFINE 'Z' VECTOR SCALE FACTOR .
1220 Medium$="BASIC/DATA_FILE/" ! DEFINE MASS STORAGE MEDIUM .
1230 !****************************************************
1240 CLEAR Hpib                         ! CLEAR HPIB BUS .
1250 WAIT 5.0
1260 PRINT CHR$(12)
1270 INPUT "ENTER DURATION OF DATA RECORD . (Seconds) ...",T_sample
1280 !*****************************
1290 !*** COMPUTE TIME BASE VECTOR ***
1300 !*****************************
1310 N_sample=INT(T_sample*F_sample)
1320 FOR I=0 TO N_sample-1
1330    T(I)=I/F_sample
1340 NEXT I
1350 INPUT "Hit ENTER when TAPE RECORDER is READY...",A$
1360 CALL Clear_cards(Hpib,Cards$)      ! CLEAR ACTIVE CARDS .
1370 FOR I=0 TO 2                       ! SET LENGTH OF SAMPLE RECORD .
1380    CALL Size_block(Hpib,N_sample,I)
1390 NEXT I
1400 CALL Arm_cards(Hpib,Cards$)        ! ARM MEMORY AND A/D CARDS .
1410 !*******************************
1420 !*** WAIT FOR END OF DATA STREAM **
1430 !*******************************
1440 WAIT 1.25*T_sample
1450 BEEP
1460 DISP "*************** READING DATA BACK THROUGH BUSS ! **************"
1470 FOR I=0 TO N_sample-1
1480    CALL Read_memory(Hpib_mi,I,X(*),Y(*),Z(*))
1490 NEXT I
1500 BEEP
1510 INPUT "Enter FILENAME of Data Stream (Omit Extension) ...",Name$
1520 Name$=Name$&".XYZ"
1530 INPUT "Enter JOB LABEL of Data Stream ...",Job$
1540 DISP "*************** SAVING DATA STREAM ON DISK MEDIUM **************"
1550 CALL Writefile3(Name$,Job$,Medium$,N_sample,X(*),Y(*),Z(*))
1560 BEEP
1570 DISP "DATA FILE STORED UNDER FILENAME :  ";Name$
1580 END
1590 !**************************************************************
1600 !********************* SUBROUTINE CLEAR_CARDS ********************
1610 !**************************************************************
1620 !*         THIS  SUBROUTINE  CLEARS  THE  MULTIPROGRAMMER  CARDS IN THE *
1630 !* SLOT NUMBERS SPECIFIED BY ' Cards$ ' .                             *
```

```
1640 !******************************************************************
1650 SUB Clear_cards(Hpib,Cards$)
1660 OUTPUT Hpib;"CC,"&Cards$&"T"
1670 SUBEND
1680 !******************************************************************
1690 !************************ SUBROUTINE ARM_CARD ***********************
1700 !******************************************************************
1710 !*     . THIS SUBROUTINE ARMS THE INTERRUPT FLAG ON THE SPECIFIED CARD. *
1720 !******************************************************************
1730 SUB Arm_cards(Hpib,Cards$)
1740 OUTPUT Hpib;"AC,"&Cards$&"T"
1750 SUBEND
1760 !******************************************************************
1770 !********************** SUBROUTINE SIZE_BLOCK **********************
1780 !******************************************************************
1790 !*          THIS SUBROUTINE DETERMINES THE LENGTH OF THE DATA VECTOR TO *
1800 !* BE SAMPLED BY THE MULTIPROGRAMMER MEMORY  CARD  BEFORE IT CEASES TO *
1810 !* TAKE FURTHER DATA . THE CARD IS SPECIFIED BY ITS SLOT NUMBER  GIVEN *
1820 !* IN THE VARIABLE ' Card_no ' .                                      *
1830 !******************************************************************
1840 SUB Size_block(Hpib,N_sample,Card_no)
1850 N_sample$=VAL$(INT(N_sample))
1860 Card_no$=VAL$(INT(2*Card_no+1))
1870 OUTPUT Hpib;"WF,"&Card_no$&".0,"&N_sample$&"T"
1880 SUBEND
1890 !******************************************************************
1900 !********************** SUBROUTINE GO_PARALLEL *********************
1910 !******************************************************************
1920 !*        THIS SUBROUTINE INVOKES THE PARALLEL MODE OF OPERATION OF THE *
1930 !* MULTIPROGRAMMMER .                                                  *
1940 !******************************************************************
1950 SUB Go_parallel(Hpib)
1960 OUTPUT Hpib;"GP"
1970 SUBEND
1980 !******************************************************************
1990 !******************** SUBROUTINE READ_MEMORY ***********************
2000 !******************************************************************
2010 !*       THIS SUBROUTINE READS A DATA SAMPLE FROM MEMORY AND RETURNS IT *
2020 !* IN THE VARIABLE ' Dummy ' .                                        *
2030 !******************************************************************
2040 SUB Read_memory(Hpib_mi,I_index,X(*),Y(*),Z(*))
2050 COM /Read_memory/ X_scale,Y_scale,Z_scale
2060 FOR J_vector=0 TO 2
2070   J_vector$=VAL$(INT(2*J_vector))
2080   OUTPUT Hpib_mi;"MI,"&J_vector$&",1T"  ! ISSUE MEMORY INPUT COMMAND .
2090   ENTER Hpib_mi;Dummy                     ! READ IN DATA WORD.
2100   SELECT J_vector
2110   CASE =0
2120      X(I_index)=X_scale*Dummy
2130   CASE =1
2140      Y(I_index)=Y_scale*Dummy
2150   CASE =2
2160      Z(I_index)=Z_scale*Dummy
2170   END SELECT
2180 NEXT J_vector
2190 SUBEND
2200 !******************************************************************
2210 !********************** SUBROUTINE WRITEFILE3 **********************
2220 !******************************************************************
2230 !*       THIS SUBROUTINE ACCEPTS THREE DATA VECTORS OF EQUAL LENGTH AND *
2240 !* WRITES  THEM  TO  A  DISK STORAGE FILE UNDER THE FILENAME SPECIFIED *
2250 !* BY THE USER .                                                       *
2260 !******************************************************************
2270 SUB Writefile3(Name$,Job$,Medium$,N_data,X(*),Y(*),Z(*))
2280 DIM File_name$[40]
2290 !**********************************************************
```

```
2300 !******************* DEFINITION OF VARIABLES *******************
2310 !***************************************************************
2320 ! Name$          ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
2330 ! Job$           ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
2340 ! Medium$        ! ADDRESS OF MASS STORAGE MEDIUM
2350 ! N_data         !-NUMBER OF DATA ELEMENTS IN EACH VECTOR .
2360 !***************************************************************
2370 !*********************************
2380 !* CREATE DATA FILE FOR STORAGE **
2390 !*********************************
2400 File_size=(N_data/9)
2410 IF Medium$=":INTERNAL" THEN
2420    File_name$=Name$&Medium$
2430 ELSE
2440    File_name$=Medium$&Name$
2450 END IF
2460 CREATE BDAT File_name$,File_size
2470 !*********************************
2480 ! ASSIGN BUFFER I/O PATH TO FILE *
2490 !*********************************
2500 ASSIGN @Path_1 TO File_name$
2510 !*********************************
2520 !** CORRECTLY SIZE DATA VECTOR ***
2530 ,*********************************
2540 REDIM X(N_data-1),Y(N_data-1),Z(N_data-1)
2550 !*********************************
2560 !******** STORE JOB LABEL ********
2570 !****************************** .**
2580 OUTPUT @Path_1;Job$
2590 !*********************************
2600 !***** STORE NUMBER OF ELEMENTS ***
2610 !*********************************
2620 OUTPUT @Path_1;N_data
2630 !*********************************
2640 !******** STORE DATA ARRAY ********
2650 !*********************************
2660 OUTPUT @Path_1;X(*),Y(*),Z(*)
2670 !*********************************
2680 !***** CLOSE FILE AND BUFFER *****
2690 !*********************************
2700 ASSIGN @Path_1 TO *
2710 SUBEND
```

```
31 Aug 1987          20:22:40

  1000 !***********************************************************************
  1010 !*************************** PROGRAM ANGLER ****************************
  1020 !***********************************************************************
  1030 !*         THIS PROGRAM  CONVERTS  THE SPECIFIED RAW XYZ WAVE COMPUTER *
  1040 !* DATA FILE INTO THE ELEVATIONAL ANGLE AZIMUTHAL ANGLE AND SEA HEIGHT *
  1050 !* FORMAT . THIS FORMAT IS DENOTED BY THE `_ANG' FILENAME EXTENSION .  *
  1060 !***********************************************************************
  1070 DIM X(5000),Y(5000),Z(5000),Phi(5000),Theta(5000)
  1080 DIM File_xyz$[16],File_ang$[16],Job$[80],Medium$[20]
  1090 Pie=4*ATN(1)
  1100 RAD
  1110 !***************************************************************
  1120 !*************** DEFINITION OF PROGRAM VARIABLES ***************
  1130 !***************************************************************
  1140 ! X(*)            ! RAW X VECTOR OUTPUT FROM WAVE COMPUTER.  (Feet)
  1150 ! Y(*)            ! RAW Y VECTOR OUTPUT FROM WAVE COMPUTER.  (Feet)
  1160 ! Z(*)            ! SEA SURFACE DISTANCE FROM BEAM SCREEN .  (Feet)
  1170 ! Phi(*)          ! SPHERICAL ELEVATIONAL ANGLE OF NORMAL.(Radians)
  1180 ! Theta(*)        ! SPHERICAL AZIMUTHAL ANGLE IF NORMAL . (Radians)
  1190 ! N_data          ! NUMBER OF DATA POINTS IN DATA VECTORS.
  1200 ! File_xyz$       ! FILENAME OF SOURCE XYZ FORMAT DATA FILE .
  1210 ! File_ang$       ! FILENAME OF SOURCE ANGLE FORMAT DATA FILE .
  1220 Medium$="BASIC/DATA_FILE/" ! DEFINE MASS STORAGE MEDIUM .
  1230 !***************************************************************
  1240 PRINT CHR$(12)
  1250 INPUT "Enter FILENAME of SOURCE XYZ DATA FILE (Omit Extension) ...",File_
xyz$
  1260 File_ang$=File_xyz$&"_ANG"
  1270 File_xyz$=File_xyz$&"_XYZ"
  1280 DISP "************* READING SOURCE FILE *************"
  1290 CALL Readfile3(File_xyz$,Job$,Medium$,N_data,X(*),Y(*),Z(*))
  1300 DISP "********* COMPUTING UNIT NORMAL ANGLES ********"
  1310 FOR I=0 TO N_data-1
  1320   !************************************
  1330   !* COMPUTE NORMAL ELEVATIONAL ANGLE *
  1340   !************************************
  1350   Phi(I)=ATN(SQR(X(I)^2+Y(I)^2)/Z(I))/2
  1360   !************************************
  1370   !** COMPUTE NORMAL AZIMUTHAL ANGLE **
  1380   !************************************
  1390   CALL Quad_just(X(I),Y(I),Pie,Dummy)
  1400   Theta(I)=Dummy-Pie
  1410 NEXT I
  1420 DISP "************ SAVING CONVERTED DATA FILE ************"
  1430 CALL Writefile3(File_ang$,Job$,Medium$,N_data,Phi(*),Theta(*),Z(*))
  1440 BEEP
  1450 DISP "FILE STORED UNDER FILENAME  :  ";File_ang$
  1460 END
  1470 !***********************************************************************
  1480 !************************ SUBROUTINE QUAD_JUST *************************
  1490 !***********************************************************************
  1500 !*         THIS  SUBROUTINE  COMPUTES  THE INVERSE TANGENT OF TWO GIVEN *
  1510 !* X AND Y COORDINATES  AND  RETURNS THE ANGLE CORRECTED TO THE PROPER *
  1520 !* QUADRANT .                                                          *
  1530 !***********************************************************************
  1540 SUB Quad_just(X,Y,Pie,Angle)
  1550 IF X<>0 THEN                    ! COMPUTE BASE ANGLE WRT X AXIS .
  1560   Angle=ATN(ABS(Y/X))
  1570 ELSE
  1580   Angle=Pie/2
  1590 END IF
  1600 IF X>=0 THEN                    ! QUADRANT COMPENSATED FOR HERE.
  1610   IF Y>=0 THEN
  1620     Angle=Angle                 ! QUADRANT I   HERE
```

```
1630   ELSE
1640      Angle=2*Pie-Angle          ! QUADRANT IV  HERE
1650   END IF
1660 ELSE                            ! X<0 BELOW !!
1670   IF Y>=0 THEN
1680      Angle=Pie-Angle            ! QUADRANT II  HERE
1690   ELSE
1700      Angle=Angle+Pie            ! QUADRANT III HERE
1710   END IF
1720 END IF
1730 SUBEND
1740 !*******************************************************************
1750 !*********************** SUBROUTINE READFILE3 ***********************
1760 !*******************************************************************
1770 !*        THIS SUBROUTINE READS THREE DATA VECTORS FROM DISK STORAGE OF *
1780 !* EQUAL LENGTH AND BOOTS THEM INTO THE DUMMY VECTORS X(*),Y(*),Z(*).  *
1790 !*******************************************************************
1800 SUB Readfile3(Name$,Job$,Medium$,N_data,X(*),Y(*),Z(*))
1810 DIM File_name$[40]
1820 !***********************************************************
1830 !****************** DEFINITION OF VARIABLES ******************
1840 !***********************************************************
1850 ! Name$          ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
1860 ! Job$           ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
1870 ! Medium$        ! ADDRESS OF MASS STORAGE MEDIUM
1880 ! N_data         ! NUMBER OF DATA ELEMENTS IN EACH VECTOR .
1890 !***********************************************************
1900 !*********************************
1910 ! ASSIGN BUFFER I/O PATH TO FILE *
1920 !*********************************
1930 IF Medium$=":INTERNAL" THEN
1940    File_name$=Name$&Medium$
1950 ELSE
1960    File_name$=Medium$&Name$
1970 END IF
1980 ASSIGN @Path_1 TO File_name$
1990 !*********************************
2000 !******* READ  JOB LABEL *********
2010 !*********************************
2020 ENTER @Path_1;Job$
2030 !*********************************
2040 !*** ENTER NUMBER OF ELEMENTS ****
2050 !*********************************
2060 ENTER @Path_1;N_data
2070 !*********************************
2080 !** CORRECTLY SIZE DATA VECTOR ***
2090 !*********************************
2100 REDIM X(N_data-1),Y(N_data-1),Z(N_data-1)
2110 !*********************************
2120 !******* READ  DATA ARRAY ********
2130 !*********************************
2140 ENTER @Path_1;X(*),Y(*),Z(*)
2150 !*********************************
2160 !***** CLOSE FILE AND BUFFER *****
2170 !*********************************
2180 ASSIGN @Path_1 TO *
2190 SUBEND
2200 !*******************************************************************
2210 !*********************** SUBROUTINE WRITEFILE3 **********************
2220 !*******************************************************************
2230 !*       THIS SUBROUTINE ACCEPTS THREE DATA VECTORS OF EQUAL LENGTH AND *
2240 !* WRITES  THEM  TO  A  DISK STORAGE FILE UNDER THE FILENAME SPECIFIED *
2250 !* BY THE USER .                                                      *
2260 !*******************************************************************
2270 SUB Writefile3(Name$,Job$,Medium$,N_data,X(*),Y(*),Z(*))
2280 DIM File_name$[40]
```

```
2290 !****************************************************************
2300 !******************** DEFINITION OF VARIABLES ********************
2310 !****************************************************************
2320 ! Name$         ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
2330 ! Job$          ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
2340 ! Medium$       ! ADDRESS OF MASS STORAGE MEDIUM
2350 ! N_data        ! NUMBER OF DATA ELEMENTS IN EACH VECTOR .
2360 !****************************************************************
2370 !**********************************
2380 !* CREATE DATA FILE FCR STORAGE **
2390 !**********************************
2400 File_size=INT(N_data/9)
2410 IF Medium$=":INTERNAL" THEN
2420    File_name$=Name$&Medium$
2430 ELSE
2440    File_name$=Medium$&Name$
2450 END IF
2460 CREATE BDAT File_name$,File_size
2470 !**********************************
2480 ! ASSIGN BUFFER I/O PATH TO FILE *
2490 !**********************************
2500 ASSIGN @Path_1 TO File_name$
2510 !**********************************
2520 !** CORRECTLY SIZE DATA VECTOR ***
2530 !**********************************
2540 REDIM X(N_data-1),Y(N_data-1),Z(N_data-1)
2550 !**********************************
2560 !******* STORE JOB LABEL *********
2570 !**********************************
2580 OUTPUT @Path_1;Job$
2590 !**********************************
2600 !**** STORE NUMBER OF ELEMENTS ***
2610 !**********************************
2620 OUTPUT @Path_1;N_data
2630 !**********************************
2640 !******* STORE DATA ARRAY ********
2650 !**********************************
2660 OUTPUT @Path_1;X(*),Y(*),Z(*)
2670 !**********************************
2680 !***** CLUSE FILE AND BUFFER *****
2690 !**********************************
2700 ASSIGN @Path_1 TO *
2710 SUBEND
```

```
31 Aug 1987          20:23:25

1000 !************************************************************************
1010 !************************** PROGRAM PEEKER ****************************
1020 !************************************************************************
1030 !*         THIS PROGRAM BOOTS IN AN ANGLULAR FORMATTED SEA SURFACE FILE *
1040 !* AND PERMITS THE USER TO PLOT AND EXAMINE IT .                         *
1050 !************************************************************************
1060 DIM Phi(4096),Theta(4096),Time_axis(4096)
1070 DIM X(4096),Y(4096),Z(4096)
1080 DIM Name$[16],Name_in$[16],Name_spec$[16],Medium$[20],Job$[80]
1090 !************************************************************
1100 !*************** DEFINITION OF LOCAL VARIABLES ***************
1110 !************************************************************
1120 F_sample=60
1130 T_sample=1/F_sample
1140 Leakage=.1
1150 Pie=4*ATN(1)
1160 Medium$="BASIC/DATA_FILE/"
1170 !************************************************************
1180 PRINT CHR$(12)
1190 INPUT "Enter FILENAME of SOURCE Data File .....",Name$
1200 INPUT "Enter TIME LIMIT on DATA STREAM ......",T_max
1210 N_data=INT(T_max/T_sample)
1220 !****************************************
1230 !********* COMPUTE TIME BASE VECTOR *********
1240 !****************************************
1250 CALL Time_base(N_data,T_sample,Time_axis(*))
1260 Length=LEN(Name$)
1270 Test$=Name$[Length-4,Length]
1280 IF Test$="_ANG" THEN
1290    CALL Readfile3(Name$,Job$,Medium$,N_point,Phi(*),Theta(*),Z(*))
1300    CALL Time_base(N_data,T_sample,Time_axis(*))
1310    CALL Plot_file(Time_axis(*),Phi(*),N_data,0,T_max,-Pie,Pie,3,"Y")
1320    CALL Plot_file(Time_axis(*),Theta(*),N_data,0,T_max,-Pie,Pie,4,"N")
1330 ELSE
1340    CALL Readfile3(Name$,Job$,Medium$,N_point,X(*),Y(*),Z(*))
1350    CALL Plot_file(Time_axis(*),X(*),N_data,0,T_max,-6,6,2,"Y")
1360    CALL Plot_file(Time_axis(*),Y(*),N_data,0,T_max,-6,6,3,"N")
1370    CALL Plot_file(Time_axis(*),Z(*),N_data,0,T_max,-6,6,4,"N")
1380 END IF
1390 PRINT Job$
1400 PRINT
1410 PRINT "Total Record Length is ";N_point;" Points....."
1420 INPUT "HIT RETURN ...",A$
1430 GRAPHICS OFF
1440 END
1450 !*************************************************************************
1460 !*********************** SUBROUTINE TIME_BASE ***********************
1470 !*************************************************************************
1480 !*          THIS  SUBROUTINE  COMPUTES THE TIME BASE VECTOR FOR USE IN   *
1490 !* PLOTTING THE DIRECTIONAL DATA .                                        *
1500 !*************************************************************************
1510 SUB Time_base(N_data,T_sample,Time_axis(*))
1520 FOR I=0 TO N_data-1
1530   Time_axis(I)=I*T_sample
1540 NEXT I
1550 SUBEND
1560 !*************************************************************************
1570 !*********************** SUBROUTINE READFILE3 ***********************
1580 !*************************************************************************
1590 !*         THIS SUBROUTINE READS THREE DATA VECTORS FROM DISK STORAGE OF *
1600 !* EQUAL LENGTH AND BOOTS THEM INTO THE DUMMY VECTORS X(*),Y(*),Z(*).   *
1610 !*************************************************************************
1620 SUB Readfile3(Name$,Job$,Medium$,N_data,X(*),Y(*),Z(*))
1630 DIM File_name$[40]
```

```
1640 !*****************************************************************
1650 !****************** DEFINITION OF VARIABLES ******************
1660 !*****************************************************************
1670 ! Name$         ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
1680 ! Job$          ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
1690 ! Medium$       ! ADDRESS OF MASS STORAGE MEDIUM
1700 ! N_data        ! NUMBER OF DATA ELEMENTS IN EACH VECTOR .
1710 !*****************************************************************
1720 !*********************************
1730 ! ASSIGN BUFFER I/O PATH TO FILE *
1740 !*********************************
1750 IF Medium$=":INTERNAL" THEN
1760    File_name$=Name$&Medium$
1770 ELSE
1780    File_name$=Medium$&Name$
1790 END IF
1800 ASSIGN @Path_1 TO File_name$
1810 !*********************************
1820 !******* READ  JOB LABEL *********
1830 !*********************************
1840 ENTER @Path_1;Job$
1850 !*********************************
1860 !*** ENTER NUMBER OF ELEMENTS ***
1870 !*********************************
1880 ENTER @Path_1;N_data
1890 !*********************************
1900 !** CORRECTLY SIZE DATA VECTOR ***
1910 !*********************************
1920 REDIM X(N_data-1),Y(N_data-1),Z(N_data-1)
1930 !*********************************
1940 !******* READ  DATA ARRAY ********
1950 !*********************************
1960 ENTER @Path_1;X(*),Y(*),Z(*)
1970 !*********************************
1980 !***** CLOSE FILE AND BUFFER *****
1990 !*********************************
2000 ASSIGN @Path_1 TO *
2010 SUBEND
2020 !*************************************************************************
2030 !*********************** SUBROUTINE PLOT_FILE ***********************
2040 !*************************************************************************
2050 !*      THIS SUBROUTINE ACCEPTS TWO DATA VECTORS AND PLOTS ONE  VERSUS *
2060 !* THE OTHER . THE  USER  NEED  ONLY  SUPPLY  THE  LIMITS OF THE GIVEN *
2070 !* VECTORS AND THE DESIRED PLOTTING COLOR . SCALING AND AXES ARE AUTO- *
2080 !* MATICALLY PROVIDED BY THIS SUBROUTINE .                             *
2090 !*************************************************************************
2100 SUB Plot_file(Xdata(*),Ydata(*),Nplot,Xmin,Xmax,Ymin,Ymax,Penc,New$)
2110 COM /Plot_block/ Xscale,Yscale,Xoffset,Yoffset
2120 !************************************************************
2130 !************** DEFINTITION OF LOCAL VARIABLES *************
2140 !************************************************************
2150 ! Xdata(*)         ! ABSCISSA DATA VECTOR TO BE PLOTTED .
2160 ! Ydata(*)         ! ORDINATE DATA VECTOR TO BE PLOTTED .
2170 ! Nplot            ! NUMBER OF DATA POINTS IN VECTORS .
2180 ! Xmin             ! SMALLEST ELEMENT IN Xdata(*) VECTOR .
2190 ! Xmax             ! LARGEST ELEMENT IN Xdata(*) VECTOR .
2200 ! Ymin             ! SMALEST ELEMENT IN Ydata(*) VECTOR .
2210 ! Ymax             ! LARGEST ELEMENT IN Ydata(*) VECTOR .
2220 ! Penc             ! DESIRED COLOR CODE OF PLOTTING COLOR  .
2230 ! New$             ! ORDERS THE ROUTINE TO CLEAR THE GRAPHICS
2240 White=1           ! DEFINE THE COLOR CODE FOR WHITE
2250 A_color=White     ! SET AXIS COLOR WHITE
2260 Xleft=0           ! DEFINE LEFT OF SCREEN      (Plotter Units)
2270 Xrail=28          ! DEFINE X AXIS RAIL         (Plotter Units)
2280 Xcenter=64        ! X COORD CENTER SCREEN      (Plotter Units)
2290 Xright=128        ! DEFINE RIGHT SCREEN        (Plotter Units)
```

```
2300 Ybottom=0          ! DEFINE LOWER SCREEN       (Plotter Units)
2310 Yrail=16           ! DEFINE Y AXIS RAIL        (Plotter Units)
2320 Ycenter=48         ! Y COORDCENTER SCREEN      (Plotter Units)
2330 Ytop=96            ! DEFINE TOP OF SCREEN      (Plotter Units)
2340 ! X_denom          ! DENOMINATOR OF X PLOTTING SCALE FACTOR .
2350 ! Y_denom          ! DENOMINATOR OF Y PLOTTING  SCALE FACTOR
2360 !**********************************************************
2370 !*****************************************************
2380 !** CLEAR AND INITIALIZE GRAPHICS IF SPECIFIED *
2390 !*****************************************************
2400 IF New$="Y" THEN
2410   GINIT 1.5
2420   GRAPHICS ON
2430   PEN White
2440   VIEWPORT Xleft,Xright,Ybottom,Ytop
2450   FRAME
2460   !**********************************
2470   !* DRAW PROPER AXES FOR PLOTTING *
2480   !**********************************
2490   IF Xmin<0 THEN
2500      IF Ymin<0 THEN        !*******************************
2510         Xoffset=Xcenter    !*  FOUR QUAD AXES DRAWN HERE *
2520         Yoffset=Ycenter    !*******************************
2530         X_denom=Xmax
2540         Y_denom=Ymax
2550         CALL Axis_draw(Xleft,Yoffset,Xright,Yoffset,A_color,-Xmax,Xmax)
2560         CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,-Ymax,Ymax)
2570      ELSE                  !*******************************
2580         Xoffset=Xcenter    !* +/- X TYPE AXIS DRAWN HERE *
2590         Yoffset=Yrail      !*******************************
2600         X_denom=Xmax
2610         Y_denom=Ymax-Ymin
2620         CALL Axis_draw(Xleft,Yoffset,Xright,Yoffset,A_color,-Xmax,Xmax)
2630         CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,Ymin,Ymax)
2640      END IF
2650   ELSE
2660      IF Ymin<0 THEN        !*******************************
2670         Xoffset=Xrail      !* +/- Y TYPE AXIS DRAWN HERE *
2680         Yoffset=Ycenter    !*******************************
2690         X_denom=Xmax-Xmin
2700         Y_denom=Ymax-Ymin
2710         CALL Axis_draw(Xoffset,Yoffset,Xright,Yoffset,A_color,Xmin,Xm
2720         CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,-Ymax,Ym
2730         Yoffset=Ybottom
2740      ELSE                  !*******************************
2750         Xoffset=Xrail      !* + ONLY X&Y AXES DRAWN HERE *
2760         Yoffset=Ybottom    !*******************************
2770         X_denom=Xmax-Xmin
2780         Y_denom=Ymax-Ymin
2790         CALL Axis_draw(Xoffset,Yoffset,Xright,Yoffset,A_color,Xmin,Xmax)
2800         CALL Axis_draw(Xoffset,Yoffset,Xoffset,Ytop,A_color,Ymin,Ymax)
2810      END IF
2820   END IF
2830   Xscale=(Xright-Xoffset)/X_denom
2840   Yscale=(Ytop-Yoffset)/Y_denom
2850 END IF
2860 !*****************************
2870 !* DATA VECTORS PLOTTED BELOW *
2880 !*****************************
2890 PENUP
2900 CALL Scaler(Xdata(0),Ydata(0),Xmin,Xmax,Ymin,Ymax,X_plot,Y_plot)
2910 PEN Penc
2920 MOVE X_plot,Y_plot
2930 FOR I=0 TO Nplot-1
2940   CALL Scaler(Xdata(I),Ydata(I),Xmin,Xmax,Ymin,Ymax,X_plot,Y_plot)
2950   DRAW X_plot,Y_plot
```

```
2960 NEXT I
2970 SUBEND
2980 !****************************************************************
2990 !********************** SUBROUTINE AXIS_DRAW ************************
3000 !****************************************************************
3010 !*       THIS SUBROUTINE DRAWS AN AXIS FROM THE STARTING COORDINATE TO *
3020 !* THE FINAL ONE . IT ALSO QUANTIFIES THE ORIGIN AND TERMINUS OF  SAID *
3030 !* AXIS .                                                             *
3040 !****************************************************************
3050 SUB Axis_draw(Xstart,Ystart,Xfinal,Yfinal,Axis_color,A_min,A_max)
3060 Pie=4*ATN(1)
3070 Delta=5
3080 PENUP
3090 PEN Axis_color
3100 PENUP
3110 MOVE Xstart,Ystart
3120 DRAW Xfinal,Yfinal
3130 PENUP
3140 CSIZE 3.0,.5
3150 CALL Rounder(A_min,3,A0)
3160 CALL Rounder(A_max,3,A1)
3170 IF Xstart=Xfinal THEN
3180   CALL Labelit(Xstart-Delta,Ystart,Pie/2,Axis_color,VAL$(A0))
3190   CALL Labelit(Xfinal-Delta,Yfinal-2*Delta,Pie/2,Axis_color,VAL$(A1))
3200 ELSE
3210   CALL Labelit(Xstart,Ystart-Delta,0,Axis_color,VAL$(A0))
3220   CALL Labelit(Xfinal-2*Delta,Ystart-Delta,0,Axis_color,VAL$(A1))
3230 END IF
3240 SUBEND
3250 !****************************************************************
3260 !********************* SUBROUTINE LABELIT ***************************
3270 !****************************************************************
3280 !* THIS SUBROUTINE SIMPLY ACCEPTS THE GIVEN LABEL AND PLACES IT WHERE   *
3290 !* IT IS SPECIFIED (ie X,Y LOCATION) AT THE GIVEN TILT ANGLE . THE PEN  *
3300 !* COLOR 'Penc' IS ALSO  PROVIDED  BY  THE  USER  . THIS SAVES A LOT OF *
3310 !* REPETITIVE CODE .                                                    *
3320 !****************************************************************
3330 SUB Labelit(X,Y,Tilt,Penc,Strng$)
3340 PENUP
3350 MOVE X,Y
3360 PEN Penc
3370 LDIR Tilt
3380 LABEL Strng$
3390 PENUP
3400 SUBEND
3410 !****************************************************************
3420 !************************ SUBROUTINE SCALER **************************
3430 !****************************************************************
3440 !*       THIS SUBROUTINE SCALES THE DATA PASSED TO IT FOR CRT PLOTTING  *
3450 !* PURPOSES .                                                           *
3460 !****************************************************************
3470 SUB Scaler(X_data,Y_data,Xmin,Xmax,Ymin,Ymax,X_plot,Y_plot)
3480 COM /Plot_block/ Xscale,Yscale,Xoffset,Yoffset
3490 X_plot=Xscale*(X_data-Xmin)+Xoffset
3500 Y_plot=Yscale*(Y_data-Ymin)+Yoffset
3510 SUBEND
3520 !****************************************************************
3530 !*********************** SUBROUTINE ROUNDER *************************
3540 !****************************************************************
3550 !*       THIS  SUBROUTINE  ACCEPTS  A  NUMBER  OF ANY SIZE OR SIGN AND *
3560 !* ROUNDS IT TO THE SPECIFIED NUMBER OF DIGITS .                        *
3570 !****************************************************************
3580 SUB Rounder(X_input,N_digits,X_rounded)
3590 !**************************************************
3600 !*********** DEFINITION OF LOCAL VARIABLES *************
3610 !**************************************************
```

```
3620 ! X_input        ! INPUT NUMBER TO BE ROUNDED
3630 ! X_dummy        ! DUMMY VARIABLE USED TO PROTECT X_input
3640 ! N_digits       ! NUMBER OF DIGITS DISPLAYED AFTER ROUNDING
3650 ! X_rounded      ! ROUNDED EQUIVALENT OF X_input
3660 ! Sign           ! NUMERICAL POLARITY OF ROUNDED NUMBER
3670 ! Magnitude      ! ORDER OF MAGNITUDE OF INPUT NUMBER
3680 ! Mantissa       ! MANTISSA OF NUMBER UNDER ROUNDING
3690 ! ARGUMENT       ! ABBREVIATED VERSION OF MANTISSA.
3700 !**********************************************************
3710 IF X_input<>0 THEN
3720   X_dummy=X_input
3730   Sign=SGN(X_dummy)
3740   X_dummy=ABS(X_dummy)
3750   Magnitude=INT(LGT(X_dummy))
3760   Mantissa=X_dummy/(10^Magnitude)
3770   Argument=INT(Mantissa*10^(N_digits-1))/10^(N_digits-1)
3. 80   X_rounded=Sign*Argument*10^Magnitude
3790 ELSE
3800   X_rounded=X_input
3810 END IF
3820 SUBEND
```

Appendix D

FOURIER TRANSFORM OPERATIONAL SOFTWARE

(1) Z_SPECTRUM Program

(2) SPEC_DERIV Program

(3) DIR_FFT Program

(4) SEE_SPEC Program

```
31 Aug 1987         20:29:49

   1000 !********************************************************************
   1010 !************************* PROGRAM Z_SPECTRUM ***************************
   1020 !********************************************************************
   1030 !*          THIS PROGRAM  BOOTS  IN  THE 'Z' VECTOR  FROM  THE ANGULAR *
   1040 !* FORMATTED  FILE  AND  PERFORMS A FAST FOURIER TRANSFORM ON IT . THE *
   1050 !* RESULTING  SPECTRUM  IS THEN  PLOTTED , PRINTED OUT , AND STORED ON *
   1060 !* DISK WITH THE FILE EXTENSION '_SPEC ' .                            *
   1070 !********************************************************************
   1080 DIM Phi(4096),Theta(4096),Z(4096)
   1090 DIM Frequency(4096),Magnitude(4096),Phase(4096)
   1100 DIM Name$[16],Name_in$[16],Name_out$[16],Medium$[20],Job$[80]
   1110 PRINT CHR$(12)
   1120 !**********************************************************
   1130 !*************** DEFINITION OF LOCAL VARIABLES ***************
   1140 !**********************************************************
   1150 ! Z(*)            ! ELEVATION OF SEA SURFACE.              (Feet)
   1160 ! Phi(*)          ! ELEVATIONAL ANGLE OF UNIT NORMAL. (Radians)
   1170 ! Theta(*)        ! AZIMUTHAL ANGLE OF UNIT NORMAL.   (Radians)
   1180 ! Magnitude(*)    ! MAGNITUDE OF FOURIER SPECTRUM.    (ft/hertz)
   1190 ! Phase(*)        ! PHASE OF FOURIER SPECTRUM.        (Radians)
   1200 ! Frequency(*)    ! FREQUENCY OF SPECTRAL COMPONENT.     (Hertz)
   1210 ! N_data          ! NUMBER OF TEMPORAL DATA POINTS.
   1220 ! N_point         ! N_data ROUNDED UP TO NEXT POWER OF TWO.
   1230 ! Mean            ! STATISTICAL MEAN OF Z VECTOR DATA.    (Feet)
   1240 F_sample=60       ! SAMPLING RATE OF WAVE COMPUTER.      (Hertz)
   1250 T_sample=1/F_sample          ! TEMPORAL SAMPLING INTERVAL.(Sec)
   1260 Pie=4*ATN(1)
   1270 Medium$="BASIC/DATA_FILE/" ! DEFINTION OF MASS STORAGE MEDIUM.
   1280 Pen1=2
   1290 Pen2=3
   1300 !**********************************************************
   1310 PRINT CHR$(12)
   1320 INPUT "Enter FILENAME of SOURCE Data File ..(Omit Extension)...",Name$
   1330 INPUT "Enter SPECTRAL TRUNCATION Data File LENGTH ...",N_short
   1340 Window_$="N"
   1350 Name_in$=Name$&"_ANG"
   1360 Name_out$=Name$&"_SPEC"
   1370 CALL Readfile3(Name_in$,Job$,Medium$,N_data,Phi(*),Theta(*),Z(*))
   1380 N_point=2^INT(LOG(N_data)/LOG(2)+1)
   1390 CALL Statistics(Z(*),N_data,Mean,Sigma)
   1400 IF Window_$="Y" THEN
   1410    CALL Windower(Z(*),N_point)
   1420 END IF
   1430 CALL Kill_offset(Z(*),N_data,Mean)
   1440 CALL Zero_fill(Z(*),N_data,N_point)
   1450 CALL Fft(Z(*),N_point,Pie,Magnitude(*),Phase(*))
   1460 CALL Freq_base(N_point,N_short,F_sample,Frequency(*))
   1470 Freq_max=MAX(Frequency(*))
   1480 Mag_max=MAX(Magnitude(*))
   1490 Phase_min=ABS(MIN(Phase(*)))
   1500 Phase_max=ABS(MAX(Phase(*)))
   1510 IF Phase_max<Phase_min THEN
   1520    Phase_max=Phase_min
   1530 END IF
   1540 PRINT CHR$(12)
   1550 PRINT Job$
   1560 PRINTER IS 6
   1570 PRINT Job$
   1580 PRINTER IS 1
   1590 CALL Plot_file(Frequency(*),Magnitude(*),N_short,0,Freq_max,-Mag_max,Mag_
max,Pen1,"Y","MAGNITUDE PLOT")
   1600 INPUT "Hit RETURN to CONTINUE ....",A$
   1610 CALL Plot_file(Frequency(*),Phase(*),N_short,0,Freq_max,-Phase_max,Phase_
max,Pen2,"Y","PHASE PLOT ")
```

```
 1620 INPUT "Hit RETURN to CONTINUE ...",A$
 1630 GRAPHICS OFF
 1640 INPUT "PRINT-OUT SPECTRUM ? (Y/N) ....",A$
 1650 IF A$="Y" THEN
 1660    CALL Print_out(Frequency(*),Magnitude(*),Phase(*),N_short)
 1670 END IF
 1680 INPUT "STORE SPECTRUM on Disk ? (Y/N)",A$
 1690 IF A$="Y" THEN
 1700    CALL Writefile3(Name_out$,Job$,Medium$,N_short,Frequency(*),Magnitude(
*),Phase(*))
 1710 END IF
 1720 PRINT CHR$(12)
 1730 END
 1740 !*****************************************************************
 1750 !************************** SUBROUTINE FFT *************************
 1760 !*****************************************************************
 1770 !*        THIS  SUBROUTINE  PERFORMS  A  FAST FOURIER TRANSFORM ON THE *
 1780 !* DEPOSITED DATA VECTOR ' X_input(*) '. THE REAL PART OF THE SPECTRAL *
 1790 !* VECTOR  IS RETURNED IN THE VARIABLE ' F_real(*) ' AND THE IMAGINARY *
 1800 !* PART IS  RETURNED  IN  VARIABLE ' F_image(*) ' . IT IS IMPORTANT TO *
 1810 !* NOTE THAT , IN ORDER  FOR  THIS FFT ALGORITHM TO WORK THE NUMBER OF *
 1820 !* DATA POINTS UNDER ANALYSIS MUST BE A POWER OF TWO !!               *
 1830 !*****************************************************************
 1840 SUB Fft(X_input(*),N_point,Pie,Magnitude(*),Phase(*))
 1850 DIM Real_1(4095),Image_1(4095),Real_2(4095),Image_2(4095)
 1860 DIM P_index(2048),Q_index(2048)
 1870 REDIM Real_1(N_point-1),Image_1(N_point-1)
 1880 REDIM Real_2(N_point-1),Image_2(N_point-1)
 1890 RAD
 1900 Pie=4*ATN(1)
 1910 V_point=INT(LOG(N_point)/LOG(2))
 1920 !**************************************************
 1930 !***** ORDER DATA VECTOR FOR INPUT OF TRANSFORM ******
 1940 !**************************************************
 1950 CALL Bit_reverse(X_input(*),N_point,V_point,Real_1(*))
 1960 !******************************
 1970 !* NULL IMAGINARY INPUT VECTOR *
 1980 !******************************
 1990 FOR I=0 TO N_point/2-1
 2000   Image_1(I)=0
 2010 NEXT I
 2020 FOR I_stage=0 TO V_point-1           !  START STAGE STROBING LOOP
 2030   CALL Butterfly(N_point,V_point,I_stage,P_index(*),Q_index(*))
 2040   FOR J_butterfly=0 TO N_point/2-1    ! START BUTTERFLY STROBING LOOP .
 2050      !*************************************
 2060      !* DETERMINE BUTTERFLY BRANCH POINTS *
 2070      !*************************************
 2080      P=P_index(J_butterfly)
 2090      Q=Q_index(J_butterfly)
 2100      R_power=FNModulo(J_butterfly*2^(V_point-1-I_stage),N_point/2)
 2110      CALL Phasor(Pie,N_point,R_power,W_real,W_image)
 2120      CALL Product_complex(W_real,W_image,Real_1(Q),Image_1(Q),Dummy_real,
Dummy_image)
 2130      !**********************************
 2140      !* COMPUTE UPPER HALF OF BUTTERFLY *
 2150      !**********************************
 2160      Real_2(P)=Real_1(P)+Dummy_real
 2170      Image_2(P)=Image_1(P)+Dummy_image
 2180      !**********************************
 2190      !* COMPUTE LOWER HALF OF BUTTERFLY *
 2200      !**********************************
 2210      Real_2(Q)=Real_1(P)-Dummy_real
 2220      Image_2(Q)=Image_1(P)-Dummy_image
 2230   NEXT J_butterfly
 2240      !**********************************
 2250      !* UPDATE NEXT CYCLE SOURCE VECTOR *
```

```
2260      !*********************************
2270      MAT Real_1=Real_2
2280      MAT Image_1=Image_2
2290 NEXT I_stage
2300 !******************************************
2310 !* DETERMINE MAGNITUDE AND PHASE OF SPECTRUM *
2320 !******************************************
2330 CALL Mag_phase(Real_2(*),Image_2(*),N_point,Magnitude(*),Phase(*))
2340 SUBEND .
2350 !****************************************************************
2360 !*********************** SUBROUTINE MAG_PHASE **********************
2370 !****************************************************************
2380 !*     THIS SUBROUTINE COMPUTES THE MAGNITUDE AND PHASE OF THE COMPLEX *
2390 !* VECTORS PROVIDED IN THE VARIABLES ' X_real(*) ' AND ' X_image(*) '. *
2400 !* THE RESULTING MAGNITUDE IS THEN STORED IN THE VECTOR   ' R_mag(*) ' *
2410 !* AND THE PHASE IS STORED IN THE VECTOR ' P_phase(*) ' .              *
2420 !****************************************************************
2430 SUB Mag_phase(X_real(*),X_image(*),N_point,R_mag(*),P_phase(*))
2440 Pie=4*ATN(1)
2450 FOR I=0 TO N_point-1
2460   R_mag(I)=SQR(X_real(I)*X_real(I)+X_image(I)*X_image(I))
2470   IF X_real(I)<>0 THEN
2480      Phase=ATN(ABS(X_image(I)/X_real(I)))
2490   ELSE
2500      Phase=Pie/2
2510   END IF
2520   X_sign=SGN(X_real(I))
2530   Y_sign=SGN(X_image(I))
2540   IF Y_sign>=0 THEN
2550      IF X_sign>=0 THEN
2560         P_phase(I)=Phase
2570      ELSE
2580         P_phase(I)=Pie-Phase
2590      END IF
2600   ELSE
2610      IF X_sign>=0 THEN
2620         P_phase(I)=-Phase
2630      ELSE
2640         P_phase(I)=Phase-Pie
2650      END IF
2660   END IF
2670 NEXT I
2680 SUBEND
2690 !****************************************************************
2700 !********************** SUBROUTINE BIT_REVERSE **********************
2710 !****************************************************************
2720 !*       THIS  SUBROUTINE  PERFORMS  A  BIT-REVERSAL  OPERATION ON THE *
2730 !* DEPOSITED INPUT VECTORS INDICES . THIS  IS  IN  PREPARATION  FOR AN *
2740 !* IN-PLACE FAST FOURIER TRANSFORM OPERATION .                         *
2750 !****************************************************************
2760 SUB Bit_reverse(Vector_in(*),N_vector,N_power,Vector_out(*))
2770 DIM Index_in(16),Index_out(16)
2780 !****************************************************
2790 !************** DEFINITION OF LOCAL VARIABLES *************
2800 !****************************************************
2810 ! Vector_in(*)    ! INPUT VECTOR TO BE BIT REVERSE SORTED.
2820 ! N_power         ! LOG BASE TWO OF INPUT VECTOR LENGTH .
2830 ! N_vector        ! ACTUAL LENGTH OF INPUT VECTOR .
2840 ! Index_in(*)     ! BINARY INPUT VECTOR REFERENCE INDEX .
2850 ! Index_out(*)    ! BINARY BIT REVERSED OUTPUT VECTOR INDEX
2860 ! Vector_out(*)   ! BIT REVERSE SORTED OUTPUT VECTOR .
2870 !****************************************************
2880 FOR I=0 TO N_power     ! NULL BIT INDEX WORDS
2890   Index_in(I)=0
2900   Index_out(I)=0
2910 NEXT I
```

```
2920 FOR I=0 TO N_vector-1
2930   IF I<>0 THEN
2940      CALL Inc_binary(Index_in(*),N_power)
2950   END IF
2960   CALL Reflect(Index_in(*),N_power,Index_out(*))
2970   CALL Base_ten(Index_in(*),N_power,I_input)
2980   CALL Base_ten(Index_out(*),N_power,I_output)
2990   !****************************************
3000   !** BIT REVERSED INDICES OPERATION BELOW .**
3010   !****************************************
3020   Vector_out(I_output)=Vector_in(I_input)
3030 NEXT I
3040 SUBEND
3050 !******************************************************************
3060 !*********************** SUBROUTINE INC_BINARY *******************
3070 !******************************************************************
3080 !*         THIS SUBROUTINE PERFORMS A BINARY INCREMENT OPERATION ON THE *
3090 !* DEPOSITED BINARY VECTOR  ' Word_inc(*) '  AND RETURNS THE RESULT IN *
3100 !* THE SAME VARIABLE .                                                 *
3110 !******************************************************************
3120 SUB Inc_binary(Word_inc(*),N_power)
3130 Carry_flag=0
3140 Done_flag=0
3150 I=0
3160 WHILE Done_flag=0
3170   IF I=0 THEN
3180      IF Word_inc(I)=0 THEN
3190         Word_inc(I)=1
3200         Done_flag=1
3210      ELSE
3220         Word_inc(I)=0
3230         Carry_flag=1
3240      END IF
3250   ELSE
3260      IF Carry_flag=1 THEN
3270         IF Word_inc(I)=0 THEN
3280            Word_inc(I)=1
3290            Done_flag=1
3300         ELSE
3310            Word_inc(I)=0
3320            Carry_flag=1
3330         END IF
3340      END IF
3350  END IF
3360  I=I+1
3370  IF I=N_power THEN
3380     Done_flag=1
3390  END IF
3400 END WHILE
3410 SUBEND
3420 !********************************************************************
3430 !************************ SUBROUTINE REFLECT ************************
3440 !********************************************************************
3450 !*     THIS SUBROUTINE TRANSPOSES THE POSITION OF THE BITS IN THE INPUT *
3460 !* VECTOR  TO  OPPOSITE  POSITIONS  WITH RESPECT TO THE CENTROID OF THE *
3470 !* BINARY WORD .                                                        *
3480 !********************************************************************
3490 SUB Reflect(Word_in(*),N_power,Word_out(*))
3500 FOR I=0 TO N_power-1
3510  Word_out(I)=Word_in(N_power-I-1)
3520 NEXT I
3530 SUBEND
3540 !******************************************************************
3550 !*********************** SUBROUTINE BASE_TEN *********************
3560 !******************************************************************
3570 !*      THIS SUBROUTINE CONVERTS THE DEPOSITED BINARY VECTOR TO A BASE *
```

7 ✓

```
3580 !* TEN INTEGER.THE BASE TEN NUMBER IS RETURNED IN THE VARIABLE 'X_out'.*
3590 !*******************************************************************
3600 SUB Base_ten(Word_in(*),N_power,X_out)
3610 X_out=0
3620 FOR I=0 TO N_power-1
3630  X_out=X_out+Word_in(I)*2^I
3640 NEXT I
3650 SUBEND
3660 !********************************************************************
3670 !********************** SUBROUTINE BUTTERFLY ***********************
3680 !*******************************************************************
3690 !*      THIS SUBROUTINE GENERATES THE NECESSARY  INDICES  DEFINING THE *
3700 !* BUTTERFLYS  WHICH  PERFORM  THE  IN-PLACE  COMPUTATIONS  OF  A FAST *
3710 !* FOURIER TRANSFORM .                                                *
3720 !*******************************************************************
3730 SUB Butterfly(N_point,V_point,Stage,P(*),Q(*))
3740 !*******************************************************
3750 !************** DEFINITION OF LOCAL VARIABLES ***************
3760 !*******************************************************
3770 ! N_point      ! NUMBER OF POINTS IN FOURIER TRANSFORM .
3780 ! V_point      ! LOG BASE TWO OF NUMBER OF TRANSFORM POINTS.
3790 ! Stage        ! STAGE OF TRANSFORM VECTOR PROCESSING .
3800 ! Span         ! WIDTH OF ROW SPAN OF BUTTERFLY .
3810 ! N_butterfly  ! NUMBER OF BUTTERFLYS IN TRANSFORM STAGE.
3820 ! N_cross      ! NUMBER OF BUTTERFLYS FOUND .
3830 ! Up_cross     ! POSITION OF UPPER BUTTERFLY BRANCH.
3840 ! Low_cross    ! POSITION OF LOWER BUTTERFLY BRANCH .
3850 ! P(*)         ! 'P' INDEX OF BUTTERFLY 'N_cross' .
3860 ! Q(*)         ! 'Q' INDEX OF BUTTERFLY 'N_cross' .
3870 !******************************************************
3880 Span=2^Stage
3890 !***************************
3900 !* DEFINE INITIAL BUTTERFLY *
3910 !***************************
3920 Up_cross=0
3930 Low_cross=Span
3940 N_cross=-1
3950 IF Span>1 THEN                    ! TEST OUT CASE OF STAGE ZERO
3960   WHILE N_cross<N_point/2-Span
3970     FOR I=Up_cross TO Low_cross-1
3980       N_cross=N_cross+1
3990       P(N_cross)=I
4000       Q(N_cross)=I+Span
4010     NEXT I
4020     Up_cross=Q(N_cross)+1
4030     Low_cross=Up_cross+Span
4040   END WHILE
4050 ELSE
4060   FOR I=0 TO N_point/2-1
4070     P(I)=2*I
4080     Q(I)=2*I+1
4090   NEXT I
4100 END IF
4110 SUBEND
4120 !*******************************************************************
4130 !************************ FUNCTION MODULO *************************
4140 !*******************************************************************
4150 !*       THIS  FUNCTION   RETURNS    THE  MODULO VALUE  OF AN  INTEGER *
4160 !* ARGUMENT WRT THE MODULO LIMIT SPECIFIED AS ' Mod_max' .            *
4170 !*******************************************************************
4180 DEF FNModulo(Number,Mod_max)
4190 Dummy=INT(Number/Mod_max)
4200 N_mod=Number-Dummy*Mod_max
4210 RETURN N_mod
4220 FNEND
4230 !*******************************************************************
```

```
4240 !********************** SUBROUTINE PRODUCT_COMPLEX **********************
4250 !*************************************************************************
4260 !*  THIS SUBROUTINE PERFORMS A COMPLEX MULTIPLICATION OPERATION ON THE *
4270 !* DEPOSITED  ' X_real + X_image '   AND   ' Y_real + Y_image '  INPUT *
4280 !* VARIABLES    AND    RETURNS    THE    RESULT    IN    THE    VARIABLES *
4290 !* ' Z_real + Z_image ' .                                                 *
4300 !*************************************************************************
4310 SUB Product_complex(X_real,X_image,Y_real,Y_image,Z_real,Z_image)
4320 !************************************************************
4330 !*************** DEFINITION OF LOCAL VARIABLES ***************
4340 !************************************************************
4350 ! X_real    ! REAL PART OF FIRST INPUT VARIABLE .
4360 ! X_image   ! IMAGINARY PART OF FIRST INPUT VARIABLE.
4370 ! Y_real    ! REAL PART OF SECOND INPUT VARIABLE .
4380 ! Y_image   ! IMAGINARY PART OF SECOND INPUT VARIABLE
4390 ! Z_real    ! REAL PART OF THE PRODUCT OF  INPUT VARIABLES .
4400 ! Z_image   ! IMAGINARY PART OF PRODUCT SUM OF INPUT VARIABLES
4410 !************************************************************
4420 Z_real=X_real*Y_real-(X_image*Y_image)
4430 Z_image=X_real*Y_image+X_image*Y_real
4440 SUBEND
4450 !*************************************************************************
4460 !************************** SUBROUTINE PHASOR ***************************
4470 !*************************************************************************
4480 !*        THIS SUBROUTINE  COMPUTES  THE REAL AND IMAGINARY PARTS OF AN *
4490 !* EXPONENTIAL UNIT TRANSFORM PHASOR RAISED TO THE POWER ' R_power '.  *
4500 !*************************************************************************
4510 SUB Phasor(Pie,N,R,W_real,W_image)
4520 W_real=COS(2*Pie*R/N)
4530 W_image=SIN(2*Pie*R/N)
4540 SUBEND
4550 !*************************************************************************
4560 !************************* SUBROUTINE ZERO_FILL *************************
4570 !*************************************************************************
4580 !*         THIS SUBROUTINE EXTENDS THE LENGTH OF THE RECORD TO THE NEXT *
4590 !* HIGHEST POWER OF TWO BY FILLING THE REMAINDER WITH ZEROES .          *
4600 !*************************************************************************
4610 SUB Zero_fill(Dummy(*),N_in,N_out)
4620 V_in=INT(LOG(N_in)/LOG(2))          ! COMPUTE POWER OF TWO OF DATA RECORD.
4630 N_out=2^(V_in+1)                    ! INCREASE RECORD LENGTH TO NEXT HIGH-
4640 REDIM Dummy(N_out-1)                ! EST PWER OF TWO .
4650 FOR I=N_in TO N_out-1
4660   Dummy(I)=0                        ! ZERO FILL REMAINDER OF DATA RECORD .
4670 NEXT I
4680 SUBEND
4690 !*************************************************************************
4700 !************************* SUBROUTINE FREQ_BASE *************************
4710 !*************************************************************************
4720 !*         THIS  SUBROUTINE  COMPUTES THE FREQUENCY BASE VECTOR FOR THE *
4730 !* RESULTANT OF THE DIRECTIONAL FOURIER TRANSFORM .                     *
4740 !*************************************************************************
4750 SUB Freq_base(N_point,N_frequency,F_sample,Frequency(*))
4760 F_delta=F_sample/N_point
4770 FOR I=0 TO N_frequency-1
4780   Frequency(I)=I*F_delta
4790 NEXT I
4800 SUBEND
4810 !*************************************************************************
4820 !************************ SUBROUTINE KILL_OFFSET ************************
4830 !*************************************************************************
4840 !*         THIS SUBROUTINE ELIMINATES THE DC OFFSET FROM THE SUPPLIED  *
4850 !* VECTOR BY SUBTRACTING ITS MEAN AND THEN NEGATES THE RESULTANT .     *
4860 !*************************************************************************
4870 SUB Kill_offset(Vector(*),N_vector,Mean)
4880 FOR I=0 TO N_vector-1
4890   Vector(I)=Mean-Vector(I)
```

```
4900 NEXT I
4910 SUBEND
4920 !*********************************************************************
4930 !*********************** SUBROUTINE READFILE3 ************************
4940 !*********************************************************************
4950 !*        THIS SUBROUTINE READS THREE DATA VECTORS FROM DISK STORAGE OF *
4960 !* EQUAL LENGTH AND BOOTS THEM INTO THE DUMMY VECTORS X(*),Y(*),Z(*).  *
4970 !*********************************************************************
4980 SUB Readfile3(Name$,Job$,Medium$,N_data,X(*),Y(*),Z(*))
4990 DIM File_name$[40]
5000 !*************************************************************
5010 !******************** DEFINITION OF VARIABLES ******************
5020 !*************************************************************
5030 ! Name$           ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
5040 ! Job$            ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
5050 ! Medium$         ! ADDRESS OF MASS STORAGE MEDIUM
5060 ! N_data          ! NUMBER OF DATA ELEMENTS IN EACH VECTOR .
5070 !*************************************************************
5080 !********************************
5090 ! ASSIGN BUFFER I/O PATH TO FILE *
5100 !********************************
5110 IF Medium$=":INTERNAL" THEN
5120    File_name$=Name$&Medium$
5130 ELSE
5140    File_name$=Medium$&Name$
5150 END IF
5160 ASSIGN @Path_1 TO File_name$
5170 !********************************
5180 !******* READ  JOB LABEL ********
5190 !********************************
5200 ENTER @Path_1;Job$
5210 !********************************
5220 !*** ENTER NUMBER OF ELEMENTS ****
5230 !********************************
5240 ENTER @Path_1;N_data
5250 !********************************
5260 !** CORRECTLY SIZE DATA VECTOR ***
5270 !********************************
5280 REDIM X(N_data-1),Y(N_data-1),Z(N_data-1)
5290 !********************************
5300 !******* READ  DATA ARRAY *******
5310 !********************************
5320 ENTER @Path_1;X(*),Y(*),Z(*)
5330 !********************************
5340 !***** CLOSE FILE AND BUFFER *****
5350 !********************************
5360 ASSIGN @Path_1 TO *
5370 REDIM X(4096),Y(4096),Z(4096)
5380 SUBEND
5390 !**********************************************************************
5400 !*********************** SUBROUTINE WRITEFILE3 ************************
5410 !**********************************************************************
5420 !*       THIS SUBROUTINE ACCEPTS THREE DATA VECTORS OF EQUAL LENGTH AND *
5430 !* WRITES  THEM  TO  A  DISK STORAGE FILE UNDER THE FILENAME SPECIFIED *
5440 !* BY THE USER .                                                       *
5450 !**********************************************************************
5460 SUB Writefile3(Name$,Job$,Medium$,N_data,X(*),Y(*),Z(*))
5470 DIM File_name$[40]
5480 !*************************************************************
5490 !******************** DEFINITION OF VARIABLES ******************
5500 !*************************************************************
5510 ! Name$           ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
5520 ! Job$            ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
5530 ! Medium$         ! ADDRESS OF MASS STORAGE MEDIUM
5540 ! N_data          ! NUMBER OF DATA ELEMENTS IN EACH VECTOR .
5550 !*************************************************************
```

```
5560 !*********************************
5570 !* CREATE DATA FILE FOR STORAGE **
5580 !*********************************
5590 File_size=INT(N_data/9)
5600 IF Medium$=":INTERNAL" THEN
5610    File_name$=Name$&Medium$
5620 ELSE
5630    File_name$=Medium$&Name$
5640 END IF
5650 CREATE BDAT File_name$,File_size
5660 !*********************************
5670 ! ASSIGN BUFFER I/O PATH TO FILE *
5680 !*********************************
5690 ASSIGN @Path_1 TO File_name$
5700 !*********************************
5710 !** CORRECTLY SIZE DATA VECTOR ***
5720 !*********************************
5730 REDIM X(N_data-1),Y(N_data-1),Z(N_data-1)
5740 !*********************************
5750 !******** STORE JOB LABEL *********
5760 !*********************************
5770 OUTPUT @Path_1;Job$
5780 !*********************************
5790 !**** STORE NUMBER OF ELEMENTS ***
5800 !*********************************
5810 OUTPUT @Path_1;N_data
5820 !*********************************
5830 !******** STORE DATA ARRAY ********
5840 !*********************************
5850 OUTPUT @Path_1;X(*),Y(*),Z(*)
5860 !*********************************
5870 !***** CLOSE FILE AND BUFFER *****
5880 !*********************************
5890 ASSIGN @Path_1 TO *
5900 SUBEND
5910 !**********************************************************************
5920 !*********************** SUBROUTINE PRINT_OUT ***********************
5930 !**********************************************************************
5940 !*            THIS  SUBROUTINE  PRINTS  OUT  THE  BEARING AND RELATIVE *
5950 !* CONTRIBUTION  FOR  EACH  FREQUENCY  COMPONENT  IN  THE DIRECTIONAL *
5960 !* SPECTRUM .                                                         *
5970 !**********************************************************************
5980 SUB Print_out(Frequency(*),Magnitude(*),Phase(*),N_data)
5990 Pie=4*ATN(1)
6000 PRINTER IS 6
6010 PRINT CHR$(12)
6020 PRINT "*******************************************************************
*************"
6030 PRINT "************************ FREQUENCY MAGNITUDES AND  PHASE *********
*************"
6040 PRINT "*******************************************************************
**************"
6050 PRINT
6060 PRINT
6070 PRINT "Frequency                Magnitude                   Phase (Degrees) "
6080 PRINT
6090 FOR I=0 TO N_data-1
6100    PRINT USING Format_1;Frequency(I),Magnitude(I),Phase(I)*180/Pie
6110 Format_1:   IMAGE DD.DDD,16X,D.DDDE,22X,SDDD.D
6120 NEXT I
6130 PRINT CHR$(12)
6140 PRINTER IS 1
6150 SUBEND
6160 !*********************************************************************
6170 !*********************** SUBROUTINE PLOT_FILE ************************
6180 !*********************************************************************
```

```
6190 !*      THIS SUBROUTINE ACCEPTS TWO DATA VECTORS AND PLOTS ONE  VERSUS *
6200 !* THE OTHER . THE  USER  NEED  ONLY  SUPPLY  THE  LIMITS OF THE GIVEN *
6210 !* VECTORS AND THE DESIRED PLOTTING COLOR . SCALING AND AXES ARE AUTO- *
6220 !* MATICALLY PROVIDED BY THIS SUBROUTINE .                             *
6230 !**********************************************************************
6240 SUB Plot_file(Xdata(*),Ydata(*),Nplot,Xmin,Xmax,Ymin,Ymax,Penc,New$,Lbl$)
6250 COM /Plot_block/ Xscale,Yscale,Xoffset,Yoffset
6260 !**********************************************************
6270 !************** DEFINTITION OF LOCAL VARIADLES *************
6280 !**********************************************************
6290 ! Xdata(*)       ! ABSCISSA DATA VECTOR TO BE PLOTTED .
6300 ! Ydata(*)       ! ORDINATE DATA VECTOR TO BE PLOTTED .
6310 ! Nplot          ! NUMBER OF DATA POINTS IN VECTORS .
6320 ! Xmin           ! SMALLEST ELEMENT IN Xdata(*) VECTOR .
6330 ! Xmax           ! LARGEST ELEMENT IN Xdata(*) VECTOR .
6340 ! Ymin           ! SMALLEST ELEMENT IN Ydata(*) VECTOR .
6350 ! Ymax           ! LARGEST ELEMENT IN Ydata(*) VECTOR .
6360 ! Penc           ; DESIRED COLOR CODE OF PLOTTING COLOR  .
6370 ! New$           ! ORDERS THE ROUTINE TO CLEAR THE GRAPHICS
6380 White=1         ! DEFINE THE COLOR CODE FOR WHITE
6390 A_color=White   ! SET AXIS COLOR WHITE
6400 Xleft=0         ! DEFINE LEFT OF SCREEN     (Plotter Units)
6410 Xrail=20        ! DEFINE X AXIS RAIL        (Plotter Units)
6420 Xcenter=64      ! X COORD CENTER SCREEN     (Plotter Units)
6430 Xright=128      ! DEFINE RIGHT SCREEN       (Plotter Units)
6440 Ybottom=0       ! DEFINE LOWER SCREEN       (Plotter Units)
6450 Yrail=16        ! DEFINE Y AXIS RAIL        (Plotter Units)
6460 Ycenter=48      ! Y COORDCENTER SCREEN      (Plotter Units)
6470 Ytop=96         ! DEFINE TOP OF SCREEN      (Plotter Units)
6480 ! X_denom       ! DENOMINATOR OF X PLOTTING SCALE FACTOR .
6490 ! Y_denom       ! DENOMINATOR OF Y PLOTTING  SCALE FACTOR
6500 !**********************************************************
6510 !**********************************************
6520 !** CLEAR AND INITIALIZE GRAPHICS IF SPECIFIED *
6530 !**********************************************
6540 IF New$="Y" THEN
6550   GINIT 1.5
6560   GRAPHICS ON
6570   PEN White
6580   VIEWPORT Xleft,Xright,Ybottom,Ytop
6590   FRAME
6600   !*********************************
6610   !* DRAW PROPER AXES FOR PLOTTING *
6620   !*********************************
6630   IF Xmin<0 THEN
6640      IF Ymin<0 THEN          !******************************
6650         Xoffset=Xcenter     !*  FOUR QUAD AXES DRAWN HERE *
6660         Yoffset=Ycenter     !******************************
6670         X_denom=Xmax
6680         Y_denom=Ymax
6690         CALL Axis_draw(Xleft,Yoffset,Xright,Yoffset,A_color,-Xmax,Xmax)
6700         CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,-Ymax,Ymax)
6710      ELSE                    !******************************
6720         Xoffset=Xcenter     !* +/- X TYPE AXIS DRAWN HERE *
6730         Yoffset=Yrail       !******************************
6740         X_denom=Xmax
6750         Y_denom=Ymax-Ymin
6760         CALL Axis_draw(Xleft,Yoffset,Xright,Yoffset,A_color,-Xmax,Xmax)
6770         CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,Ymin,Ymax)
6780      END IF
6790   ELSE
6800      IF Ymin<0 THEN         !******************************
6810         Xoffset=Xrail      !* +/- Y TYPE AXIS DRAWN HERE *
6820         Yoffset=Ycenter    !******************************
6830         X_denom=Xmax-Xmin
6840         Y_denom=Ymax-Ymin
```

```
6850        CALL Axis_draw(Xoffset,Yoffset,Xright,Yoffset,A_color,Xmin,Xmax)
6860        CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,-Ymax,Ymax)
6870        Yoffset=Ybottom
6880     ELSE            !*****************************
6890        Xoffset=Xrail    !* * ONLY X&Y AXES DRAWN HERE *
6900        Yoffset=Ybottom  !*****************************
6910        X_denom=Xmax-Xmin
6920        Y_denom=Ymax-Ymin
6930        CALL Axis_draw(Xoffset,Yoffset,Xright,Yoffset,A_color,Xmin,Xmax)
6940        CALL Axis_draw(Xoffset,Yoffset,Xoffset,Ytop,A_color,Ymin,Ymax)
6950     END IF
6960   END IF
6970   Xscale=(Xright-Xoffset)/X_denom
6980   Yscale=(Ytop-Yoffset)/Y_denom
6990 END IF
7000 !****************************
7010 !* DATA VECTORS PLOTTED BELOW *
7020 !****************************
7030 PENUP
7040 CALL Scaler(Xdata(0),Ydata(0),Xmin,Xmax,Ymin,Ymax,X_plot,Y_plot)
7050 PEN Penc
7060 MOVE X_plot,Y_plot
7070 FOR I=0 TO Nplot-1
7080   CALL Scaler(Xdata(I),Ydata(I),Xmin,Xmax,Ymin,Ymax,X_plot,Y_plot)
7090   DRAW X_plot,Y_plot
7100 NEXT I
7110 MOVE Xcenter,Ybottom
7120 LDIR 0
7130 LABEL Lbl$
7140 SUBEND
7150 !*****************************************************************
7160 !********************** SUBROUTINE AXIS_DRAW ***********************
7170 !*****************************************************************
7180 !*       THIS SUBROUTINE DRAWS AN AXIS FROM THE STARTING COORDINATE TO *
7190 !* THE FINAL ONE . IT ALSO QUANTIFIES THE ORIGIN AND TERMINUS OF  SAID *
7200 !* AXIS .                                                          *
7210 !*****************************************************************
7220 SUB Axis_draw(Xstart,Ystart,Xfinal,Yfinal,Axis_color,A_min,A_max)
7230 Pie=4*ATN(1)
7240 Delta=5
7250 PENUP
7260 PEN Axis_color
7270 PENUP
7280 MOVE Xstart,Ystart
7290 DRAW Xfinal,Yfinal
7300 PENUP
7310 CSIZE 3.0,.5
7320 CALL Rounder(A_min,3,A0)
7330 CALL Rounder(A_max,3,A1)
7340 IF Xstart=Xfinal THEN
7350   CALL Labelit(Xstart-Delta,Ystart,Pie/2,Axis_color,VAL$(A0))
7360   CALL Labelit(Xfinal-Delta,Yfinal-2*Delta,Pie/2,Axis_color,VAL$(A1))
7370 ELSE
7380   CALL Labelit(Xstart,Ystart-Delta,0,Axis_color,VAL$(A0))
7390   CALL Labelit(Xfinal-2*Delta,Ystart-Delta,0,Axis_color,VAL$(A1))
7400 END IF
7410 SUBEND
7420 !******************************************************************
7430 !********************** SUBROUTINE LABELIT **************************
7440 !******************************************************************
7450 !* THIS SUBROUTINE SIMPLY ACCEPTS THE GIVEN LABEL AND PLACES IT WHERE   *
7460 !* IT IS SPECIFIED (ie X,Y LOCATION) AT THE GIVEN TILT ANGLE . THE PEN  *
7470 !* COLOR 'Penc' IS ALSO  PROVIDED  BY  THE  USER  . THIS SAVES A LOT OF *
7480 !* REPETITIVE CODE .                                                  *
7490 !******************************************************************
7500 SUB Labelit(X,Y,Tilt,Penc,Strng$)
```

```
7510 PENUP
7520 MOVE X,Y
7530 PEN Penc
7540 LDIR Tilt
7550 LABEL Strng$
7560 PENUP
7570 SUBEND
7580 !***************************************************************
7590 !************************ SUBROUTINE SCALER **********************
7600 !***************************************************************
7610 !*     .THIS SUBROUTINE SCALES THE DATA PASSED TO IT FOR CRT PLOTTING  *
7620 !* PURPOSES .                                                          *
7630 !***************************************************************
7640 SUB Scaler(X_data,Y_data,Xmin,Xmax,Ymin,Ymax,X_plot,Y_plot)
7650 COM /Plot_block/ Xscale,Yscale,Xoffset,Yoffset
7660 X_plot=Xscale*(X_data-Xmin)+Xoffset
7670 Y_plot=Yscale*(Y_data-Ymin)+Yoffset
7680 SUBEND
7690 !***************************************************************
7700 !*********************** SUBROUTINE ROUNDER ***********************
7710 !***************************************************************
7720 !*        THIS  SUBROUTINE  ACCEPTS  A  NUMBER  OF ANY SIZE OR SIGN AND *
7730 !* ROUNDS IT TO THE SPECIFIED NUMBER OF DIGITS .                       *
7740 !***************************************************************
7750 SUB Rounder(X_input,N_digits,X_rounded)
7760 !**********************************************
7770 !************ DEFINITION OF LOCAL VARIABLES ************
7780 !**********************************************
7790 ! X_input       ! INPUT NUMBER TO BE ROUNDED
7800 ! X_dummy       ! DUMMY VARIABLE USED TO PROTECT X_input
7810 ! N_digits      ! NUMBER OF DIGITS DISPLAYED AFTER ROUNDING
7820 ! X_rounded     ! ROUNDED EQUIVALENT OF X_input
7830 ! Sign          ! NUMERICAL POLARITY OF ROUNDED NUMBER
7840 ! Magnitude     ! ORDER OF MAGNITUDE OF INPUT NUMBER
7850 ! Mantissa      ! MANTISSA OF NUMBER UNDER ROUNDING
7860 ! ARGUMENT      ! ABBRIEVIATED VERSION OF MANTISSA.
7870 !**********************************************
7880 IF X_input<>0 THEN
7890   X_dummy=X_input
7900   Sign=SGN(X_dummy)
7910   X_dummy=ABS(X_dummy)
7920   Magnitude=INT(LGT(X_dummy))
7930   Mantissa=X_dummy/(10^Magnitude)
7940   Argument=INT(Mantissa*10^(N_digits-1))/10^(N_digits-1)
7950   X_rounded=Sign*Argument*10^Magnitude
7960 ELSE
7970   X_rounded=X_input
7980 END IF
7990 SUBEND
8000 !***************************************************************
8010 !*********************** SUBROUTINE STATISTICS *********************
8020 !***************************************************************
8030 !*        THIS  SUBROUTINE PROVIDES THE  MEAN AND VARIANCE  OF A RANDOM *
8040 !* VECTOR  DEPOSITED  IN  THE  VARIABLE ' Vector(*) ' .  THE  MEAN  IS *
8050 !* DEPOSITED IN THE VARIABLE ' Mean ' AND THE VARIANCE IN THE VARIABLE *
8060 !* ' Variance ' .                                                      *
8070 !***************************************************************
8080 SUB Statistics(Vector(*),N_vector,Mean,Variance)
8090 !**************************************************
8100 !*************** DEFINITION OF LOCAL VARIABLES ***************
8110 !**************************************************
8120 ! Vector(*)       ! PSEUDORANDOM VECTOR TO ANALYZE .
8130 ! N_Vector        ! LENGTH OF PSUEDORANDOM VECTOR .
8140 ! Mean            ! STATISTICAL AVERAGE OF RANDOM VECTOR .
8150 ! Variance        ! MEAN SQUARED AVERAGE DEVIATION FROM MEAN .
8160 ! Sum_vector      ! DUMMY INTEGRATION VARIABLE .
```

```
8170 !****************************************************************
8180 !****************************
8190 !* DETERMINE MEAN OF VECTOR *
8200 !****************************
8210 Sum_vector=0
8220 FOR I=0 TO N_vector-1
8230   Suu_vector=Sum_vector+Vector(I)
8240 NEXT I
8250 Mean=Sum_vector/N_vecsor
8260 !********************************
8270 !* DETERMINE VARIANCE OF VECTOR *
8280 !********************************
8290 Sum_vector=0
8300 FOR I=0 TO N_vector-1
8310   Sum_vector=Sum_vector+(Vector(I)-Mean)^2
8320 NEXT I
8330 Variance=Sum_vector/N_vector
8340 SUBEND
8350  !*********************************************************************
8360  !************************ SUBROUTINE WINDOWER ************************
8370  !*********************************************************************
8380  !*          THIS SUBROUTINE PERFORMS A TRIANGULAR WINDOWINC OPERATION *
8390  !* ON THE SUPPLIED VECTOR . THIS IS A PRELUDE TO A FOURIER TRANSFORM *
8400  !* OPERATION AND IS INTENDED TO REDUCE SPECTRAL SIDE LOBING .        *
8410  !*********************************************************************
8420  SUB Windower(Vector(*),N_vector)
8430  FOR I=0 TO N_vector-1
8440     IF I<N_vector/2 THEN
8450        Vector(I)=Vector(I)*(2*I/N_vector)
8460     ELSE
8470        Vector(I)=Vector(I)*(1-2*(I-N_vector/2)/N_vector)
8480     END IF
8490  NEXT I
8500  SUBEND
```

```
22 Sep 1987         22:57:26

1000  !********************************************************************
1010  !************************ PROGRAM SPEC_DERIV ************************
1020  !********************************************************************
1030  !*          THIS PROGRAM GENERATES AND COMPUTES THE FOURIER TRANSFORM *
1040  !* OF THE SPATIAL DERIVATIVES dZ/dx AND dZ/dy AND WRITES THE RESULT- *
1050  !* ANT TO DISK AND THE PRINTER IF REQUESTED .                        *
1060  !********************************************************************
1070  DIM Zdx_mag(4096),Zdy_mag(4096),Zdx_sigh(4096),Zdy_sigh(4096)
1080  DIM Name$[16],Name_ang$[16],Name_sdx$[16],Name_sdy$[16],Job$[80]
1090  DIM Phi(4096),Theta(4096),Dz_dx(4096),Dz_dy(4096)
1100  DIM Frequency(4096),Dummy(4096)
1110  PRINT CHR$(12)
1120  !**********************************************************
1130  !************* DEFINITION OF PROGRAM VARIABLES ************
1140  !**********************************************************
1150  ! Phi(*)        ! ELEVATIONAL ANGLE OF UNIT NORMAL. (Radians)
1160  ! Theta(*)      ! AZIMUTHAL ANGLE OF UNIT NORMAL.   (Radians)
1170  ! Dummy(*)      ! DUMMY VARIABLE USED TO ACCEPT Z(*).
1180  ! Dz_dx(*)      ! PARTIAL DERIVATIVE OF Z WRT X.(Ratiometric)
1190  ! Dz_dy(*)      ! PARTIAL DERIVATIVE OF Z WRT Y.(Ratiometric)
1200  ! Zdx_mag(*)    ! TRANSFORMED SPECTRAL MAGNITUDE OF Dz_dx(*).
1210  ! Zdy_mag(*)    ! TRANSFORMED SPECTRAL MAGNITUDE OF Dz_dy(*).
1220  ! Zdx_sigh(*)   ! TRANSFORMED SPECTRAL PHASE OF Dz_dx(*).
1230  ! Zdy_sigh(*)   ! TRANSFORMED SPECTRAL PHASE OF DZ_dy(*).
1240  ! N_data        ! DATA POINT NUMBER IN TEMPORAL DATA STREAM.
1250  ! N_point       ! N_data ROUNDED UP TO NEXT POWER OF TWO.
1260  Medium$="BASIC/DATA_FILE/"
1270  Pie=4*ATN(1)
1280  T_sample=1/60  ! TEMPORAL SAMPLING INTERVAL.      (Seconds)
1290  F_sample=60    ! SAMPLING FREQUENCY.                (Hertz)
1300  ! Name_ang$     ! ANGULAR FORMATTED DATA FILE NAME.
1310  ! Name_sdx$     ! FILENAME OF FOURIER TRANSFORMED Dz_dx.
1320  ! Name_sdy$     ! FILENAME OF FOURIER TRANSFORMED Dz_dy.
1330  !**********************************************************
1340  INPUT "Enter FILENAME of SOURCE DATA (Omit Extension) ....",Name$
1350  INPUT "Enter SPECTRAL TRUNCATION LENGTH .....",N_short
1360  Window_$="N"
1370  Name_ang$=Name$&"_ANG"
1380  Name_sdx$=Name$&"_SDX"
1390  Name_sdy$=Name$&"_SDY"
1400  !**********************************************
1410  !* BOOT IN DIRECTIONAL AND VERTICAL SPECTRAL DATA *
1420  !**********************************************
1430  CALL Readfile3(Name_ang$,Job$,Medium$,N_data,Phi(*),Theta(*),Dummy(*))
1440  N_point=2^INT(LOG(N_data)/LOG(2)+1)
1450  !**********************************************
1460  !******* COMPUTE SPATIAL DERIVATIVE VECTORS *******
1470  !**********************************************
1480  CALL Make_slopes(Phi(*),Theta(*),N_data,Dz_dx(*),Dz_dy(*))
1490  !**********************************************
1500  !* COMPUTE SPECTRUM OF SPATIAL DERIVATIVE VECTORS *
1510  !**********************************************
1520  DISP "****** COMPUTING SPATIAL DERIVATIVE FOURIER TRANSFORMS ******"
1530  IF Window_$="Y" THEN
1540     CALL Windower(Dz_dx(*),N_point)
1550     CALL Windower(Dz_dy(*),N_point)
1560  END IF
1570  CALL Fft(Dz_dx(*),N_point,Pie,Zdx_mag(*),Zdx_sigh(*))
1580  CALL Fft(Dz_dy(*),N_point,Pie,Zdy_mag(*),Zdy_sigh(*))
1590  CALL Freq_base(N_point,N_short,F_sample,Frequency(*))
1600  !**********************************************
1610  !********* OUTPUT DATA TO DISK AND PRINTER ********
1620  !**********************************************
1630  INPUT "STORE Spectral Vectors on DISK ? (Y/N)....",A$
```

```
 1640  IF A$="Y" THEN
 1650      CALL Writefile3(Name_sdx$,Job$,Medium$,N_short,Frequency(*),Zdx_mag(*
),Zdx_sigh(*))
 1660      CALL Writefile3(Name_sdy$,Job$,Medium$,N_short,Frequency(*),Zdy_mag(*
),Zdy_sigh(*))
 1670  END IF
 1680  INPUT "PRINT-OUT Spectral Components ......",A$
 1690  IF A$="Y" THEN
 1700      CALL Print_out5(Frequency(*),Zdx_mag(*),Zdx_sigh(*),Zdy_mag(*),Zdy_si
gh(*),N_short)
 1710  END IF
 1720  END
 1730 !*****************************************************************
 1740 !************************** SUBROUTINE FFT ***************************
 1750 !*****************************************************************
 1760 !*        THIS  SUBROUTINE  PERFORMS  A  FAST FOURIER TRANSFORM ON THE *
 1770 !* DEPOSITED DATA VECTOR ' X_input(*) '. THE REAL PART OF THE SPECTRAL *
 1780 !* VECTOR  IS RETURNED IN THE VARIABLE ' F_real(*) ' AND THE IMAGINARY *
 1790 !* PART IS  RETURNED  IN  VARIABLE ' F_image(*) ' . IT IS IMPORTANT TO *
 1800 !* NOTE THAT , IN ORDER  FOR  THIS FFT ALGORITHM TO WORK THE NUMBER OF *
 1810 !* DATA POINTS UNDER ANALYSIS MUST BE A POWER OF TWO !!                *
 1820 !*****************************************************************
 1830 SUB Fft(X_input(*),N_point,Pie,Magnitude(*),Phase(*))
 1840 DIM Real_1(4096),Image_1(4096),Real_2(4096),Image_2(4096)
 1850 DIM P_index(2048),Q_index(2048)
 1860 REDIM Real_1(N_point-1),Image_1(N_point-1)
 1870 REDIM Real_2(N_point-1),Image_2(N_point-1)
 1880 RAD
 1890 Pie=4*ATN(1)
 1900 V_point=INT(LOG(N_point)/LOG(2))
 1910 !***************************************************
 1920 !***** ORDER DATA VECTOR FOR INPUT OF TRANSFORM ******
 1930 !***************************************************
 1940 CALL Bit_reverse(X_input(*),N_point,V_point,Real_1(*))
 1950 !*****************************
 1960 !* NULL IMAGINARY INPUT VECTOR *
 1970 !*****************************
 1980 FOR I=0 TO N_point/2-1
 1990   Image_1(I)=0
 2000 NEXT I
 2010 FOR I_stage=0 TO V_point-1             !  START STAGE STROBING LOOP
 2020   CALL Butterfly(N_point,V_point,I_stage,P_index(*),Q_index(*))
 2030   FOR J_butterfly=0 TO N_point/2-1     ! START BUTTERFLY STROBING LOOP .
 2040       !************************************
 2050       !* DETERMINE BUTTERFLY BRANCH POINTS *
 2060       !************************************
 2070       P=P_index(J_butterfly)
 2080       Q=Q_index(J_butterfly)
 2090       R_power=FNModulo(J_butterfly*2^(V_point-1-I_stage),N_point/2)
 2100       CALL Phasor(Pie,N_point,R_power,W_real,W_image)
 2110       CALL Product_complex(W_real,W_image,Real_1(Q),Image_1(Q),Dummy_real,
Dummy_image)
 2120       !**********************************
 2130       !* COMPUTE UPPER HALF OF BUTTERFLY *
 2140       !**********************************
 2150       Real_2(P)=Real_1(P)+Dummy_real
 2160       Image_2(P)=Image_1(P)+Dummy_image
 2170       !**********************************
 2180       !* COMPUTE LOWER HALF OF BUTTERFLY *
 2190       !**********************************
 2200       Real_2(Q)=Real_1(P)-Dummy_real
 2210       Image_2(Q)=Image_1(P)-Dummy_image
 2220   NEXT J_butterfly
 2230       !**********************************
 2240       !* UPDATE NEXT CYCLE SOURCE VECTOR *
 2250       !**********************************
```

```
2260      MAT Real_1=Real_2
2270      MAT Image_1=Image_2
2280 NEXT I_stage
2290 !********************************************
2300 !* DETERMINE MAGNITUDE AND PHASE OF SPECTRUM *
2310 !********************************************
2320 CALL Mag_phase(Real_2(*),Image_2(*),N_point,Magnitude(*),Phase(*))
2330 SUBEND
2340 !*****************************************************************
2350 !*********************** SUBROUTINE MAG_PHASE *********************
2360 !*****************************************************************
2370 !*     THIS SUBROUTINE COMPUTES THE MAGNITUDE AND PHASE OF THE COMPLEX *
2380 !* VECTORS PROVIDED IN THE VARIABLES ' X_real(*) ' AND ' X_image(*) '. *
2390 !* THE RESULTING MAGNITUDE IS THEN STORED IN THE VECTOR  ' R_mag(*) ' *
2400 !* AND THE PHASE IS STORED IN THE VECTOR ' P_phase(*) ' .            *
2410 !*****************************************************************
2420 SUB Mag_phase(X_real(*),X_image(*),N_point,R_mag(*),P_phase(*))
2430 Pie=4*ATN(1)
2440 FOR I=0 TO N_point-1
2450   R_mag(I)=SQR(X_real(I)*X_real(I)+X_image(I)*X_image(I))
2460   IF X_real(I)<>0 THEN
2470      Phase=ATN(ABS(X_image(I)/X_real(I)))
2480   ELSE
2490      Phase=Pie/2
2500   END IF
2510   X_sign=SGN(X_real(I))
2520   Y_sign=SGN(X_image(I))
2530   IF Y_sign>=0 THEN
2540      IF X_sign>=0 THEN
2550         P_phase(I)=Phase
2560      ELSE
2570         P_phase(I)=Pie-Phase
2580      END IF
2590   ELSE
2600      IF X_sign>=0 THEN
2610         P_phase(I)=-Phase
2620      ELSE
2630         P_phase(I)=Phase-Pie
2640      END IF
2650   END IF
2660 NEXT I
2670 SUBEND
2680 !*****************************************************************
2690 !*********************** SUBROUTINE BIT_REVERSE *******************
2700 !*****************************************************************
2710 !*     THIS  SUBROUTINE  PERFORMS  A  BIT-REVERSAL  OPERATION ON THE *
2720 !* DEPOSITED INPUT VECTORS INDICES . THIS  IS  IN  PREPARATION  FOR AN *
2730 !* IN-PLACE FAST FOURIER TRANSFORM OPERATION .                        *
2740 !*****************************************************************
2750 SUB Bit_reverse(Vector_in(*),N_vector,N_power,Vector_out(*))
2760 DIM Index_in(16),Index_out(16)
2770 !****************************************************
2780 !************* DEFINITION OF LOCAL VARIABLES ************
2790 !****************************************************
2800 ! Vector_in(*)   ! INPUT VECTOR TO BE BIT REVERSE SORTED.
2810 ! N_power        ! LOG BASE TWO OF INPUT VECTOR LENGTH .
2820 ! N_vector       ! ACTUAL LENGTH OF INPUT VECTOR .
2830 ! Index_in(*)    ! BINARY INPUT VECTOR REFERENCE INDEX .
2840 ! Index_out(*)   ! BINARY BIT REVERSED OUTPUT VECTOR INDEX
2850 ! Vector_out(*)  ! BIT REVERSE SORTED OUTPUT VECTOR .
2860 !****************************************************
2870 FOR I=0 TO N_power    ! NULL BIT INDEX WORDS
2880   Index_in(I)=0
2890   Index_out(I)=0
2900 NEXT I
2910 FOR I=0 TO N_vector-1
```

```
2920   IF I<>0 THEN
2930      CALL Inc_binary(Index_in(*),N_power)
2940   END IF
2950   CALL Reflect(Index_in(*),N_power,Index_out(*))
2960   CALL Base_ten(Index_in(*),N_power,I_input)
2970   CALL Base_ten(Index_out(*),N_power.I_output)
2980   !****************************************
2990   !** BIT REVERSED INDICES OPERATION BELOW .**
3000   !****************************************
3010   Vector_out(I_output)=Vector_in(I_input)
3020 NEXT I
3030 SUBEND
3040 !******************************************************************
3050 !********************** SUBROUTINE INC_BINARY **********************
3060 !******************************************************************
3070 !*         THIS SUBROUTINE PERFORMS A BINARY INCREMENT OPERATION ON THE *
3080 !* DEPOSITED BINARY VECTOR  ' Word_inc(*) '  AND RETURNS THE RESULT IN *
3090 !* THE SAME VARIABLE .                                                 *
3100 !******************************************************************
3110 SUB Inc_binary(Word_inc(*),N_power)
3120 Carry_flag=0
3130 Done_flag=0
3140 I=0
3150 WHILE Done_flag=0
3160   IF I=0 THEN
3170      IF Word_inc(I)=0 THEN
3180         Word_inc(I)=1
3190         Done_flag=1
3200      ELSE
3210         Word_inc(I)=0
3220         Carry_flag=1
3230      END IF
3240   ELSE
3250      IF Carry_flag=1 THEN
3260         IF Word_inc(I)=0 THEN
3270            Word_inc(I)=1
3280            Done_flag=1
3290         ELSE
3300            Word_inc(I)=0
3310            Carry_flag=1
3320         END IF
3330      END IF
3340  END IF
3350  I=I+1
3360  IF I=N_power THEN
3370     Done_flag=1
3380  END IF
3390 END WHILE
3400 SUBEND
3410 !*********************************************************************
3420 !************************ SUBROUTINE REFLECT *************************
3430 !*********************************************************************
3440 !*      THIS SUBROUTINE TRANSPOSES THE POSITION OF THE BITS IN THE INPUT *
3450 !* VECTOR  TO  OPPOSITE  POSITIONS  WITH RESPECT TO THE CENTROID OF THE *
3460 !* BINARY WORD .                                                         *
3470 !*********************************************************************
3480 SUB Reflect(Word_in(*),N_power,Word_out(*))
3490 FOR I=0 TO N_power-1
3500  Word_out(I)=Word_in(N_power-I-1)
3510 NEXT I
3520 SUBEND
3530 !********************************************************************
3540 !*********************** SUBROUTINE BASE_TEN *************************
3550 !********************************************************************
3560 !*       THIS SUBROUTINE CONVERTS THE DEPOSITED BINARY VECTOR TO A BASE *
3570 !* TEN INTEGER.THE BASE TEN NUMBER IS RETURNED IN THE VARIABLE 'X_out' *
```

```
3580 !*****************************************************************
3590 SUB Base_ten(Word_in(*),N_power,X_out)
3600 X_out=0
3610 FOR I=0 TO N_power-1
3620  X_out=X_out+Word_in(I)*2^I
3630 NEXT I
3640 SUBEND
3650 !*****************************************************************
3660 !*********************** SUBROUTINE BUTTERFLY ***********************
3670 !*****************************************************************
3680 !*       THIS SUBROUTINE GENERATES THE NECESSARY  INDICES  DEFINING THE *
3690 !* BUTTERFLIES  WHICH  PERFORM  THE IN-PLACE  COMPUTATIONS  OF  A FAST *
3700 !* FOURIER TRANSFORM .                                                 *
3710 !*****************************************************************
3720 SUB Butterfly(N_point,V_point,Stage,P(*),Q(*))
3730 !*******************************************************
3740 !************** DEFINITION OF LOCAL VARIABLES ****************
3750 !*******************************************************
3760 ! N_point      ! NUMBER OF POINTS IN FOURIER TRANSFORM .
3770 ! V_point      ! LOG BASE TWO OF NUMBER OF TRANSFORM POINTS.
3780 ! Stage        ! STAGE OF TRANSFORM VECTOR PROCESSING .
3790 ! Span         ! WIDTH OF ROW SPAN OF BUTTERFLY .
3800 ! N_butterfly  ! NUMBER OF BUTTERFLIES IN TRANSFORM STAGE.
3810 ! N_cross      ! NUMBER OF BUTTERFLIES FOUND .
3820 ! Up_cross     ! POSITION OF UPPER BUTTERFLY BRANCH.
3830 ! Low_cross    ! POSITION OF LOWER BUTTERFLY BRANCH .
3840 ! P(*)         ! 'P' INDEX OF BUTTERFLY 'N_cross' .
3850 ! Q(*)         ! 'Q' INDEX OF BUTTERFLY 'N_cross' .
3860 !******************************************************
3870 Span=2^Stage
3880 !****************************
3890 !* DEFINE INITIAL BUTTERFLY *
3900 !****************************
3910 Up_cross=0
3920 Low_cross=Span
3930 N_cross=-1
3940 IF Span>1 THEN                      ! TEST OUT CASE OF STAGE ZERO
3950   WHILE N_cross<N_point/2-Span
3960      FOR I=Up_cross TO Low_cross-1
3970         N_cross=N_cross+1
3980         P(N_cross)=I
3990         Q(N_cross)=I+Span
4000      NEXT I
4010      Up_cross=Q(N_cross)+1
4020      Low_cross=Up_cross+Span
4030   END WHILE
4040 ELSE
4050   FOR I=0 TO N_point/2-1
4060      P(I)=2*I
4070      Q(I)=2*I+1
4080   NEXT I
4090 END IF
4100 SUBEND
4110 !******************************************************************
4120 !************************* FUNCTION MODULO **************************
4130 !******************************************************************
4140 !*         THIS  FUNCTION   RETURNS    THE  MODULO VALUE  OF AN  INTEGER *
4150 !* ARGUMENT WRT THE MODULO LIMIT SPECIFIED AS ' Mod_max' .             *
4160 !******************************************************************
4170 DEF FNModulo(Number,Mod_max)
4180 Dummy=INT(Number/Mod_max)
4190 N_mod=Number-Dummy*Mod_max
4200 RETURN N_mod
4210 FNEND
4220 !******************************************************************
4230 !********************* SUBROUTINE PRODUCT_COMPLEX ********************
```

```
4240 !*****************************************************************
4250 !*  THIS SUBROUTINE PERFORMS A COMPLEX MULTIPLICATION OPERATION ON THE *
4260 !* DEPOSITED  ' X_real + X_image '   AND   ' Y_real + Y_image '  INPUT *
4270 !* VARIABLES    AND    RETURNS    THE    RESULT    IN    THE    VARIABLES *
4280 !* ' Z_real + Z_image ' .                                              *
4290 !****************************************************************
4300 SUB Product_complex(X_real,X_image,Y_real,Y_image,Z_real,Z_image)
4310 !*****************************************************
4320 !************** DEFINITION OF LOCAL VARIABLES ***************
4330 !*****************************************************
4340 ! X_real    ! REAL PART OF FIRST INPUT VARIABLE .
4350 ! X_image   ! IMAGINARY PART OF FIRST INPUT VARIABLE.
4360 ! Y_real    ! REAL PART OF SECOND INPUT VARIABLE .
4370 ! Y_image   ! IMAGINARY PART OF SECOND INPUT VARIABLE
4380 ! Z_real    ! REAL PART OF THE PRODUCT OF  INPUT VARIABLES .
4390 ! Z_image   ! IMAGINARY PART OF PRODUCT SUM OF INPUT VARIABLES
4400 !*****************************************************
4410 Z_real=X_real*Y_real-(X_image*Y_image)
4420 Z_image=X_real*Y_image+X_image*Y_real
4430 SUBEND
4440 !*****************************************************************
4450 !************************** SUBROUTINE PHASOR **************************
4460 !*****************************************************************
4470 !*        THIS SUBROUTINE  COMPUTES  THE REAL AND IMAGINARY PARTS OF AN *
4480 !* EXPONENTIAL UNIT TRANSFORM PHASOR RAISED TO THE POWER ' R_power '.  *
4490 !*****************************************************************
4500 SUB Phasor(Pie,N,R,W_real,W_image)
4510 W_real=COS(2*Pie*R/N)
4520 W_image=SIN(2*Pie*R/N)
4530 SUBEND
4540 !*****************************************************************
4550 !************************ SUBROUTINE ZERO_FILL ************************
4560 !*****************************************************************
4570 !*          THIS SUBROUTINE EXTENDS THE LENGTH OF THE RECORD TO THE NEXT *
4580 !* HIGHEST POWER OF TWO BY FILLING THE REMAINDER WITH ZEROES .         *
4590 !*****************************************************************
4600 SUB Zero_fill(Dummy(*),N_in,N_out)
4610 V_in=INT(LOG(N_in)/LOG(2))           ! COMPUTE POWER OF TWO OF DATA RECORD.
4620 N_out=2^(V_in+1)                     ! INCREASE RECORD LENGTH TO NEXT HIGH-
4630 REDIM Dummy(N_out-1)                 ! EST PWER OF TWO .
4640 FOR I=N_in TO N_out-1
4650   Dummy(I)=0                         ! ZERO FILL REMAINDER OF DATA RECORD .
4660 NEXT I
4670 SUBEND
4680 !*****************************************************************
4690 !*********************** SUBROUTINE MAKE_SLOPES ***********************
4700 !*****************************************************************
4710 !*             THIS SUBROUTINE GENERATES THE SPATIAL DERIVATIVE VECTORS  *
4720 !* FROM THE ANGULAR FORMATTED WAVE COMPUTER DATA FILES .               *
4730 !*****************************************************************
4740 SUB Make_slopes(Phi(*),Theta(*),N_data,Dz_dx(*),Dz_dy(*))
4750 FOR I=0 TO N_data-1
4760    Dz_dx(I)=-TAN(Phi(I))*COS(Theta(I))
4770    Dz_dy(I)=-TAN(Phi(I))*SIN(Theta(I))
4780 NEXT I
4790 SUBEND
4800 !******************************************************************
4810 !******************** SUBROUTINE WRITEFILE3 ************************
4820 !******************************************************************
4830 !*       THIS SUBROUTINE ACCEPTS THREE DATA VECTORS OF EQUAL LENGTH AND *
4840 !* WRITES  THEM  TO  A  DISK STORAGE FILE UNDER THE FILENAME SPECIFIED *
4850 !* BY THE USER .                                                        *
4860 !******************************************************************
4870 SUB Writefile3(Name$,Job$,Medium$,N_data,X(*),Y(*),Z(*))
4880 DIM File_name$[40]
4890 !*****************************************************
```

```
4900 !********* ********** DEFINITION OF VARIABLES ******************
4910 !****************************************************************
4920 ! Name$         ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
4930 ! Job$          ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
4940 ! Medium$       ! ADDRESS OF MASS STORAGE MEDIUM
4950 ! N_data        ! NUMBER OF DATA ELEMENTS IN EACH VECTOR .
4960 !****************************************************************
4970 !********************************
4980 !* CREATE DATA FILE FOR STORAGE **
4990 !********************************
5000 File_size=INT(2*N_data/9)
5010 IF Medium$=":INTERNAL" THEN
5020    File_name$=Name$&Medium$
5030 ELSE
5040    File_name$=Medium$&Name$
5050 END IF
5060 CREATE BDAT File_name$,File_size
5070 !********************************
5080 ! ASSIGN BUFFER I/O PATH TO FILE *
5090 !********************************
5100 ASSIGN @Path_1 TO File_name$
5110 !********************************
5120 !** CORRECTLY SIZE DATA VECTOR ***
5130 !********************************
5140 REDIM X(N_data-1),Y(N_data-1),Z(N_data-1)
5150 !********************************
5160 !******* STORE JOB LABEL *********
5170 !********************************
5180 OUTPUT @Path_1;Job$
5190 !********************************
5200 !**** STORE NUMBER OF ELEMENTS ***
5210 !********************************
5220 OUTPUT @Path_1;N_data
5230 !********************************
5240 !******* STORE DATA ARRAY ********
5250 !********************************
5260 OUTPUT @Path_1;X(*),Y(*),Z(*)
5270 !********************************
5280 !***** CLOSE FILE AND BUFFER *****
5290 !********************************
5300 ASSIGN @Path_1 TO *
5310 SUBEND
5320 !*******************************************************************
5330 !********************** SUBROUTINE READFILE3 ***********************
5340 !*******************************************************************
5350 !*       THIS SUBROUTINE READS THREE DATA VECTORS FROM DISK STORAGE OF *
5360 !* EQUAL LENGTH AND ROOTS THEM INTO THE DUMMY VECTORS X(*),Y(*),Z(*).  *
5370 !*******************************************************************
5380 SUB Readfile3(Name$,Job$,Medium$,N_data,X(*),Y(*),Z(*))
5390 DIM File_name$[40]
5400 !****************************************************************
5410 !****************** DEFINITION OF VARIABLES ******************
5420 !****************************************************************
5430 ! Name$         ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
5440 ! Job$          ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
5450 ! Medium$       ! ADDRESS OF MASS STORAGE MEDIUM
5460 ! N_data        ! NUMBER OF DATA ELEMENTS IN EACH VECTOR .
5470 !****************************************************************
5480 !********************************
5490 ! ASSIGN BUFFER I/O PATH TO FILE *
5500 !********************************
5510 IF Medium$=":INTERNAL" THEN
5520    File_name$=Name$&Medium$
5530 ELSE
5540    File_name$=Medium$&Name$
5550 END IF
```

```
5560 ASSIGN @Path_1 TO File_name$
5570 !*******************************
5580 !******* READ  JOB LABEL ********
5590 !*******************************
5600 ENTER @Path_1;Job$
5610 !*******************************
5620 !*** ENTER NUMBER OF ELEMENTS ***
5630 !*******************************
5640 ENTER @Path_1;N_data
5650 !*******************************
5660 !** CORRECTLY SIZE DATA VECTOR **
5670 !*******************************
5680 REDIM X(N_data-1),Y(N_data-1),Z(N_data-1)
5690 !*******************************
5700 !******* READ  DATA ARRAY *******
5710 !*******************************
5720 ENTER @Path_1;X(*),Y(*),Z(*)
5730 !*******************************
5740 !***** CLOSE FILE AND BUFFER ****
5750 !*******************************
5760 ASSIGN @Path_1 TO *
5770 REDIM X(4096),Y(4096),Z(4096)
5780 SUBEND
5790 !*************************************************************
5800 !********************** SUBROUTINE PRINT_OUTS *****************
5810 !*************************************************************
5820 !*      THIS  SUBROUTINE  PRINTS OUT A SIX VARIABLE DATA TABLE ON THE *
5830 !* LINE PRINTER .                                               *
5840 !*************************************************************
5850 SUB Print_out5(X1(*),X2(*),X3(*),X4(*),X5(*),N_print)
5860 PRINTER IS 6
5870 Pie=4*ATN(1)
5880 PRINT CHR$(12)
5890 PRINT
5900 PRINT
5910 PRINT "***************************************************************
************"
5920 PRINT "**************** SPECTRAL COMPONENTS OF SPATIAL DERIVATIVES *****
************"
5930 PRINT "***************************************************************
************"
5940 PRINT
5950 PRINT
5960 PRINT "Frequency     MAG F(dZ/dx)    PHASE F(dZ/dx) MAG F(dZ/dy)   PHASE
F(dZ/dy) "
5970 PRINT
5980 FOR I=0 TO N_print-1
5990    PRINT USING Format_1;X1(I),X2(I),X3(I)*180/Pie,X4(I),X5(I)*180/Pie
6000 Format_1:   IMAGE 1X,DD.DDD,8X,D.DDE,10X,SDDD.D,8X,D.DDE,7X,SDDD.D
6010 NEXT I
6020 PRINT CHR$(12)
6030 PRINTER IS 1
6040 SUBEND
6050 !*************************************************************
6060 !*********************** SUBROUTINE FREQ_BASE ****************
6070 !*************************************************************
6080 !*        THIS  SUBROUTINE  COMPUTES THE FREQUENCY BASE VECTOR FOR THE *
6090 !* RESULTANT OF THE DIRECTIONAL FOURIER TRANSFORM .              *
6100 !*************************************************************
6110 SUB Freq_base(N_point,N_frequency,F_sample,Frequency(*))
6120 F_delta=F_sample/N_point
6130 FOR I=0 TO N_frequency-1
6140   Frequency(I)=I*F_delta
6150 NEXT I
6160 SUBEND
6170 !*************************************************************
```

```
6180 !*********************** SUBROUTINE PLOT_FILE ************************
6190 !*******************************************************************
6200 !*      THIS SUBROUTINE ACCEPTS TWO DATA VECTORS AND PLOTS ONE  VERSUS *
6210 !* THE OTHER . THE  USER  NEED  ONLY  SUPPLY  THE  LIMITS OF THE GIVEN *
6220 !* VECTORS AND THE DESIRED PLOTTING COLOR . SCALING AND AXES ARE AUTO- *
6230 !* MATICALLY PROVIDED BY THIS SUBROUTINE .                             *
6240 !*******************************************************************
6250 SUB Plot_file(Xdata(*),Ydata(*),Nplot,Xmin,Xmax,Ymin,Ymax,Penc,New$)
6260 COM /Plot_block/ Xscale,Yscale,Xoffset,Yoffset
6270 !*****************************************************
6280 !************** DEFINTITION OF LOCAL VARIABLES ************
6290 !*****************************************************
6300 ! Xdata(*)        ! ABSCISSA DATA VECTOR TO BE PLOTTED .
6310 ! Ydata(*)        ! ORDINATE DATA VECTOR TO BE PLOTTED .
6320 ! Nplot           ! NUMBER OF DATA POINTS IN VECTORS .
6330 ! Xmin            ! SMALLEST ELEMENT IN Xdata(*) VECTOR .
6340 ! Xmax            ! LARGEST ELEMENT IN Xdata(*) VECTOR .
6350 ! Ymin            ! SMALLEST ELEMENT IN Ydata(*) VECTOR .
6360 ! Ymax            ! LARGEST ELEMENT IN Ydata(*) VECTOR .
6370 ! Penc            ! DESIRED COLOR CODE OF PLOTTING COLOR  .
6380 ! New$            ! ORDERS THE ROUTINE TO CLEAR THE GRAPHICS
6390 White=1          ! DEFINE THE COLOR CODE FOR WHITE
6400 A_color=White    ! SET AXIS COLOR WHITE
6410 Xleft=0          ! DEFINE LEFT OF SCREEN      (Plotter Units)
6420 Xrail=20         ! DEFINE X AXIS RAIL         (Plotter Units)
6430 Xcenter=64       ! X COORD CENTER SCREEN      (Plotter Units)
6440 Xright=128       ! DEFINE RIGHT SCREEN        (Plotter Units)
6450 Ybottom=0        ! DEFINE LOWER SCREEN        (Plotter Units)
6460 Yrail=16         ! DEFINE Y AXIS RAIL         (Plotter Units)
6470 Ycenter=48       ! Y COORDCENTER SCREEN       (Plotter Units)
6480 Ytop=96          ! DEFINE TOP OF SCREEN       (Plotter Units)
6490 ! X_denom        ! DENOMINATOR OF X PLOTTING SCALE FACTOR .
6500 ! Y_denom        ! DENOMINATOR OF Y PLOTTING  SCALE FACTOR
6510 !*****************************************************
6520 !******************************************
6530 !** CLEAR AND INITIALIZE GRAPHICS IF SPECIFIED *
6540 !******************************************
6550 IF New$="Y" THEN
6560   GINIT 1.5
6570   GRAPHICS ON
6580   PEN White
6590   VIEWPORT Xleft,Xright,Ybottom,Ytop
6600   FRAME
6610   !*******************************
6620   !* DRAW PROPER AXES FOR PLOTTING *
6630   !*******************************
6640   IF Xmin<0 THEN
6650     IF Ymin<0 THEN          !****************************
6660       Xoffset=Xcenter      !*  FOUR QUAD AXES DRAWN HERE *
6670       Yoffset=Ycenter      !****************************
6680       X_denom=Xmax
6690       Y_denom=Ymax
6700       CALL Axis_draw(Xleft,Yoffset,Xright,Yoffset,A_color,-Xmax,Xmax)
6710       CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,-Ymax,Ymax)
6720     ELSE                    !****************************
6730       Xoffset=Xcenter      !* +/- X TYPE AXIS DRAWN HERE *
6740       Yoffset=Yrail        !****************************
6750       X_denom=Xmax
6760       Y_denom=Ymax-Ymin
6770       CALL Axis_draw(Xleft,Yoffset,Xright,Yoffset,A_color,-Xmax,Xmax)
6780       CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,Ymin,Ymax)
6790     END IF
6800   ELSE
6810     IF Ymin<0 THEN         !****************************
6820       Xoffset=Xrail        !* +/- Y TYPE AXIS DRAWN HERE *
6830       Yoffset=Ycenter      !****************************
```

```
6840         X_denom=Xmax-Xmin
6850         Y_denom=Ymax-Ymin
6860         CALL Axis_draw(Xoffset,Yoffset,Xright,Yoffset,A_color,Xmin,Xmax)
6870         CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,-Ymax,Ymax)
6880         Yoffset=Ybottom
6890      ELSE               !******************************
6900         Xoffset=Xrail       !* * ONLY X&Y AXES DRAWN HERE *
6910         Yoffset=Ybottom     !******************************
6920         X_denom=Xmax-Xmin
6930         Y_denom=Ymax-Ymin
6940         CALL Axis_draw(Xoffset,Yoffset,Xright,Yoffset,A_color,Xmin,Xmax)
6950         CALL Axis_draw(Xoffset,Yoffset,Xoffset,Ytop,A_color,Ymin,Ymax)
6960      END IF
6970   END IF
6980   Xscale=(Xright-Xoffset)/X_denom
6990   Yscale=(Ytop-Yoffset)/Y_denom
7000 END IF
7010 !******************************
7020 !* DATA VECTORS PLOTTED BELOW *
7030 !******************************
7040 PENUP
7050 CALL Scaler(Xdata(0),Ydata(0),Xmin,Xmax,Ymin,Ymax,X_plot,Y_plot)
7060 PEN Penc
7070 MOVE X_plot,Y_plot
7080 FOR I=0 TO Nplot-1
7090   CALL Scaler(Xdata(I),Ydata(I),Xmin,Xmax,Ymin,Ymax,X_plot,Y_plot)
7100   DRAW X_plot,Y_plot
7110 NEXT I
7120 SUBEND
7130 !*****************************************************************
7140 !********************** SUBROUTINE AXIS_DRAW *********************
7150 !*****************************************************************
7160 !*        THIS SUBROUTINE DRAWS AN AXIS FROM THE STARTING COORDINATE TO *
7170 !* THE FINAL ONE . IT ALSO QUANTIFIES THE ORIGIN AND TERMINUS OF  SAID *
7180 !* AXIS .                                                            *
7190 !*****************************************************************
7200 SUB Axis_draw(Xstart,Ystart,Xfinal,Yfinal,Axis_color,A_min,A_max)
7210 Pie=4*ATN(1)
7220 Delta=5
7230 PENUP
7240 PEN Axis_color
7250 PENUP
7260 MOVE Xstart,Ystart
7270 DRAW Xfinal,Yfinal
7280 PENUP
7290 CSIZE 3.0,.5
7300 CALL Rounder(A_min,3,A0)
7310 CALL Rounder(A_max,3,A1)
7320 IF Xstart=Xfinal THEN
7330   CALL Labelit(Xstart-Delta,Ystart,Pie/2,Axis_color,VAL$(A0))
7340   CALL Labelit(Xfinal-Delta,Yfinal-2*Delta,Pie/2,Axis_color,VAL$(A1))
7350 ELSE
7360   CALL Labelit(Xstart,Ystart-Delta,0,Axis_color,VAL$(A0))
7370   CALL Labelit(Xfinal-2*Delta,Ystart-Delta,0,Axis_color,VAL$(A1))
7380 END IF
7390 SUBEND
7400 !*******************************************************************
7410 !********************** SUBROUTINE LABELIT ***************************
7420 !*******************************************************************
7430 !* THIS SUBROUTINE SIMPLY ACCEPTS THE GIVEN LABEL AND PLACES IT WHERE   *
7440 !* IT IS SPECIFIED (ie X,Y LOCATION) AT THE GIVEN TILT ANGLE . THE PEN  *
7450 !* COLOR 'Penc' IS ALSO  PROVIDED  BY  THE  USER  . THIS SAVES A LOT OF *
7460 !* REPETITIVE CODE .                                                    *
7470 !*******************************************************************
7480 SUB Labelit(X,Y,Tilt,Penc,Strng$)
7490 PENUP
```

```
7500 MOVE X,Y
7510 PEN Penc
7520 LDIR Tilt
7530 LABEL Strng$
7540 PENUP
7550 SUBEND
7560 !**********************************************************************
7570 !************************* SUBROUTINE SCALER **************************
7580 !**********************************************************************
7590 !*      THIS SUBROUTINE SCALES THE DATA PASSED TO IT FOR CRT PLOTTING  *
7600 !* PURPOSES .                                                          *
7610 !**********************************************************************
7620 SUB Scaler(X_data,Y_data,Xmin,Xmax,Ymin,Ymax,X_plot,Y_plot)
7630 COM /Plot_block/ Xscale,Yscale,Xoffset,Yoffset
7640 X_plot=Xscale*(X_data-Xmin)+Xoffset
7650 Y_plot=Yscale*(Y_data-Ymin)+Yoffset
7660 SUBEND
7670 !**********************************************************************
7680 !*********************** SUBROUTINE ROUNDER ***************************
7690 !**********************************************************************
7700 !*       THIS  SUBROUTINE  ACCEPTS  A  NUMBER  OF ANY SIZE OR SIGN AND *
7710 !* ROUNDS IT TO THE SPECIFIED NUMBER OF DIGITS .                       *
7720 !**********************************************************************
7730 SUB Rounder(X_input,N_digits,X_rounded)
7740 !********************************************************
7750 !************ DEFINITION OF LOCAL VARIABLES *************
7760 !********************************************************
7770 ! X_input       ! INPUT NUMBER TO BE ROUNDED
7780 ! X_dummy       ! DUMMY VARIABLE USED TO PROTECT X_input
7790 ! N_digits      ! NUMBER OF DIGITS DISPLAYED AFTER ROUNDING
7800 ! X_rounded     ! ROUNDED EQUIVALENT OF X_input
7810 ! Sign          ! NUMERICAL POLARITY OF ROUNDED NUMBER
7820 ! Magnitude     ! ORDER OF MAGNITUDE OF INPUT NUMBER
7830 ! Mantissa      ! MANTISSA OF NUMBER UNDER ROUNDING
7840 ! ARGUMENT      ! ABBREVIATED VERSION OF MANTISSA.
7850 !********************************************************
7860 IF X_input<>0 THEN
7870   X_dummy=X_input
7880   Sign=SGN(X_dummy)
7890   X_dummy=ABS(X_dummy)
7900   Magnitude=INT(LGT(X_dummy))
7910   Mantissa=X_dummy/(10^Magnitude)
7920   Argument=INT(Mantissa*10^(N_digits-1))/10^(N_digits-1)
7930   X_rounded=Sign*Argument*10^Magnitude
7940 ELSE
7950   X_rounded=X_input
7960 END IF
7970 SUBEND
7980  !**********************************************************************
7990  !*********************** SUBROUTINE WINDOWER **************************
8000  !**********************************************************************
8010  !*          THIS SUBROUTINE PERFORMS A TRIANGULAR WINDOWING OPERATION *
8020  !* ON THE SUPPLIED VECTOR . THIS IS A PRELUDE TO A FOURIER TRANSFORM *
8030  !* OPERATION AND IS INTENDED TO REDUCE SPECTRAL SIDE LOBING .         *
8040  !**********************************************************************
8050  SUB Windower(Vector(*),N_vector)
8060  FOR I=0 TO N_vector-1
8070     IF I<N_vector/2 THEN
8080        Vector(I)=Vector(I)*(2*I/N_vector)
8090     ELSE
8100        Vector(I)=Vector(I)*(1-2*(I-N_vector/2)/N_vector)
8110     END IF
8120  NEXT I
8130  SUBEND
```

```
22 Sep 1987          23:06:54

1000 !*************************************************************
1010 !************************* PROGRAM DIR_FFT *******************
1020 !*************************************************************
1030 !*         THIS PROGRAM BOOTS IN AN ANGLULAR FORMATTED SEA SURFACE FILE *
1040 !* AND PERFORMS A  DIRECTIONAL FAST FOURIER  ON THE SIN(Phi(*)) . THIS *
1050 !* OPERATION WILL TEND TO EMPHASIZE THE SHORTER WAVELENGTH COMPONENTS. *
1060 !* THE RELATIVE CONTRIBUTION FREQUENCY AND DIRECTION OF EACH COMPONENT *
1070 !* IS  THEN  IDENTIFIED FROM THE ARRAY 'Spectrum(*,*)' . THE RESULTING *
1080 !* FREQUENCY, DIRECTION  AND  RELATIVE CONTRIBUTION  IS THEN STORED ON *
1090 !* DISK FOR LATER USE .                                                 *
1100 !*************************************************************
1110 COM /Fourier/ Spectrum(512,36),Frequency(512),Direction(36)
1120 DIM Phi(4096),Theta(4096),Z(4096),Dummy(512)
1130 DIM Integrand(4096),Magnitude(4096),Phase(4096)
1140 DIM Bearing(512),Rel_cont(512)
1150 DIM Name$[16],Name_in$[16],Name_out$[16],Medium$[20],Job$[80]
1160 !*****************************************************
1170 !************** DEFINITION OF LOCAL VARIABLES **************
1180 !*****************************************************
1190 ! Phi(*)          ! ELEVATIONAL ANGLE OF UNIT NORMAL. (Radians)
1200 ! Tneta(*)        ! AZIMUTHAL ANGLE OF UNIT NORMAL.   (Radians)
1210 ! Dummy(*)        ! DUMMY VARIABLE USED TO ACCEPT Z(*).
1220 ! Spectrum(*,*)   ! UNIT NORMAL DIRECTIONAL MAGNITUDE SPECTRUM.
1230 ! Frequency(*)    ! FREQUENCY ORDINATE AXIS OF SPECTRUM.(He-tz)
1240 ! Direction(*)    ! UNIT TEST VECTOR ANGULAR BEARING. (Degrees)
1250 ! Integrand(*)    ! NORMAL PROJECTION OF TEST VECTOR.
1260 ! Magnitude(*)    ! SPECTRAL MAGNITUDE OF PROJECTION TRANSFORM.
1270 ! Phase(*)        ! PHASE SPECTRUM OF PROJECTION TRANSFORM.
1280 ! Bearing(*)      ! ANGULAR DIRECTION OF WAVE FRONT.  (Radians)
1290 ! Rel_cont(*)     ! SPECTRUM RELATIVE CONTRIBUTION OF COMPONENT
1300 ! N_data          ! NUMBER OF DATA POINTS IN TEMPORAL STREAM.
1310 ! N_point         ! N_data ROUNDED UP TO NEXT POWER OF TWO.
1320 ! N_direction     ! NUMBER OF TEST DIRECTION VECTOR STEPS.
1330 ! N_short         ! TRUNCATED SPECTRUM COMPONENT LENGTH.
1340 ! D_gamma         ! ANGULAR STEP SIZE OF TEST DIRECTION VECTOR.
1350 F_sample=60      ! WAVE COMPUTER SAMPLING FREQUENCY.   (Hertz)
1360 T_sample=1/F_sample ! TEMPORAL SAMPLING INTERVAL.    (Seconds)
1370 Pie=4*ATN(1)
1380 Medium$="BASIC/DATA_FILE/"! DEFINTION OF MASS STORAGE MEDIUM.
1390 ! Name_in$        ! ANGULAR FORMATTED SOURCE DATA FILE.
1400 ! Name_out$       ! FILENAME OF DIRECTIONAL INFORMATION FILE.
1410 !*****************************************************
1420 PRINT CHR$(12)
1430 INPUT "Enter FILENAME of SOURCE Data File . (Omit Extension)...",Name$
1440 INPUT "Enter DIRECTIONAL ANGULAR STEP SIZE (Degrees) ...",D_gamma
1450 INPUT "Enter TRUNCATED DATA STREAM LENGTH ....",N_short
1460 Window_$="N"
1470 N_direction=INT(180/D_gamma)
1480 Name_in$=Name$&"_MNG"
1490 Name_out$=Name$&"_DIR"
1500 CALL Readfile3(Name_in$,Job$,Medium$,N_data,Phi(*),Theta(*),Z(*))
1510 N_point=2^INT(LOG(N_data)/LOG(2)+1)
1520 CALL Zero_fill(Phi(*),N_data,N_point)
1530 CALL Zero_fill(Theta(*),N_data,N_point)
1540 FOR I=0 TO N_direction-1
1550     BEEP
1560     DISP "********** OPERATION ";INT(100*I/N_direction);" PERCENT DONE ****
******"
1570     Direction(I)=I*D_gamma
1580     Gamma=Direction(I)*Pie/180
1590     CALL Integrand(Phi(*),Theta(*),Pie,N_point,Gamma,Integrand(*))
1600     IF Window_$="Y" THEN
1610        CALL Windower(Integrand(*),N_point)
1620     END IF
```

```
 1630    CALL Fft(Integrand(*),N_point,Pie,Magnitude(*),Phase(*))
 1640    CALL Stuff_array(Magnitude(*),N_short,I)
 1650 NEXT I
 1660 CALL Freq_base(N_point,N_short,F_sample,Frequency(*))
 1670 CALL Find_peaks(Frequency(*),Direction(*),Bearing(*),Rel_cont(*),N_short,
N_direction)
 1680 INPUT "STORE Directional Spectrum on DISK ? (Y/N) ....",A$
 1690 IF A$="Y" THEN
 1700    CALL Writefile3(Name_out$,Job$,Medium$,N_short,Frequency(*),Bearing(*)
,Rel_cont(*))
 1710 END IF
 1720 INPUT "Dump Directional Spectrum to the PRINTER ? (Y/N)....",A$
 1730 IF A$="Y" THEN
 1740    CALL Print_out(Frequency(*),Bearing(*),Rel_cont(*),N_short)
 1750 END IF
 1760 END
 1770 !*****************************************************************
 1780 !************************** SUBROUTINE FFT ***********************
 1790 !*****************************************************************
 1800 !*        THIS  SUBROUTINE  PERFORMS  A  FAST FOURIER TRANSFORM ON THE *
 1810 !* DEPOSITED DATA VECTOR ' X_input(*) '. THE REAL PART OF THE SPECTRAL *
 1820 !* VECTOR  IS RETURNED IN THE VARIABLE ' F_real(*) ' AND THE IMAGINARY *
 1830 !* PART IS  RETURNED  IN  VARIABLE ' F_image(*) ' . IT IS IMPORTANT TO *
 1840 !* NOTE THAT , IN ORDER  FOR  THIS FFT ALGORITHM TO WORK THE NUMBER OF *
 1850 !* DATA POINTS UNDER ANALYSIS MUST BE A POWER OF TWO !!                *
 1860 !*****************************************************************
 1870 SUB Fft(X_input(*),N_point,Pie,Magnitude(*),Phase(*))
 1880 DIM Real_1(4096),Image_1(4096),Real_2(4096),Image_2(4096)
 1890 DIM P_index(2048),Q_index(2048)
 1900 REDIM Real_1(N_point-1),Image_1(N_point-1)
 1910 REDIM Real_2(N_point-1),Image_2(N_point-1)
 1920 RAD
 1930 Pie=4*ATN(1)
 1940 V_point=INT(LOG(N_point)/LOG(2))
 1950 !*********************************************
 1960 !***** ORDER DATA VECTOR FOR INPUT OF TRANSFORM ******
 1970 !*********************************************
 1980 CALL Bit_reverse(X_input(*),N_point,V_point,Real_1(*))
 1990 !*****************************
 2000 !* NULL IMAGINARY INPUT VECTOR *
 2010 !*****************************
 2020 FOR I=0 TO N_point/2-1
 2030   Image_1(I)=0
 2040 NEXT I
 2050 FOR I_stage=0 TO V_point-1            !  START STAGE STROBING LOOP
 2060   CALL Butterfly(N_point,V_point,I_stage,P_index(*),Q_index(*))
 2070   FOR J_butterfly=0 TO N_point/2-1    ! START BUTTERFLY STROBING LOOP .
 2080      !***********************************
 2090      !* DETERMINE BUTTERFLY BRANCH POINTS *
 2100      !***********************************
 2110      P=P_index(J_butterfly)
 2120      Q=Q_index(J_butterfly)
 2130      R_power=FNModulo(J_butterfly*2^(V_point-1-I_stage),N_point/2)
 2140      CALL Phasor(Pie,N_point,R_power,W_real,W_image)
 2150      CALL Product_complex(W_real,W_image,Real_1(Q),Image_1(Q),Dummy_real,
Dummy_image)
 2160      !********************************
 2170      !* COMPUTE UPPER HALF OF BUTTERFLY *
 2180      !********************************
 2190      Real_2(P)=Real_1(P)+Dummy_real
 2200      Image_2(P)=Image_1(P)+Dummy_image
 2210      !********************************
 2220      !* COMPUTE LOWER HALF OF BUTTERFLY *
 2230      !********************************
 2240      Real_2(Q)=Real_1(P)-Dummy_real
 2250      Image_2(Q)=Image_1(P)-Dummy_image
```

```
2260   NEXT J_butterfly
2270      !**********************************
2280      !* UPDATE NEXT CYCLE SOURCE VECTOR *
2290      !**********************************
2300      MAT Real_1=Real_2
2310      MAT Image_1=Image_2
2320 NEXT I_stage
2330 !*****************************************
2340 !* DETERMINE MAGNITUDE AND PHASE OF SPECTRUM *
2350 !*****************************************
2360 CALL Mag_phase(Real_2(*),Image_2(*),N_point,Magnitude(*),Phase(*))
2370 SUBEND
2380 !*************************************************************
2390 !************************ SUBROUTINE MAG_PHASE ***********************
2400 !*************************************************************
2410 !*      THIS SUBROUTINE COMPUTES THE MAGNITUDE AND PHASE OF THE COMPLEX *
2420 !* VECTORS PROVIDED IN THE VARIABLES ' X_real(*) ' AND ' X_image(*) '. *
2430 !* THE RESULTING MAGNITUDE IS THEN STORED IN THE VECTOR   ' R_mag(*) ' *
2440 !* AND THE PHASE IS STORED IN THE VECTOR ' P_phase(*) ' .            *
2450 !*************************************************************
2460 SUB Mag_phase(X_real(*),X_image(*),N_point,R_mag(*),P_phase(*))
2470 Pie=4*ATN(1)
2480 FOR I=0 TO N_point-1
2490   R_mag(I)=SQR(X_real(I)*X_real(I)+X_image(I)*X_image(I))
2500   IF X_real(I)<>0 THEN
2510      Phase=ATN(ABS(X_image(I)/X_real(I)))
2520   ELSE
2530      Phase=Pie/2
2540   END IF
2550   X_sign=SGN(X_real(I))
2560   Y_sign=SGN(X_image(I))
2570   IF Y_sign>=0 THEN
2580      IF X_sign>=0 THEN
2590         P_phase(I)=Phase
2600      ELSE
2610         P_phase(I)=Pie-Phase
2620      END IF
2630   ELSE
2640      IF X_sign>=0 THEN
2650         P_phase(I)=-Phase
2660      ELSE
2670         P_phase(I)=Phase-Pie
2680      END IF
2690   END IF
2700 NEXT I
2710 SUBEND
2720 !*************************************************************
2730 !*********************** SUBROUTINE BIT_REVERSE **********************
2740 !*************************************************************
2750 !*        THIS  SUBROUTINE  PERFORMS  A  BIT-REVERSAL  OPERATION ON THE *
2760 !* DEPOSITED INPUT VECTORS INDICES . THIS  IS  IN  PREPARATION  FOR AN *
2770 !* IN-PLACE FAST FOURIER TRANSFORM OPERATION .                        *
2780 !*************************************************************
2790 SUB Bit_reverse(Vector_in(*),N_vector,N_power,Vector_out(*))
2800 DIM Index_in(16),Index_out(16)
2810 !**********************************************
2820 !************** DEFINITION OF LOCAL VARIABLES *************
2830 !**********************************************
2840 ' Vector_in(*)    ' INPUT VECTOR TO BE BIT REVERSE SORTED.
2850 ' N_power         ' LOG BASE TWO OF INPUT VECTOR LENGTH .
2860 ' N_vector        ' ACTUAL LENGTH OF INPUT VECTOR .
2870 ' Index_in(*)     ' BINARY INPUT VECTOR REFERENCE INDEX .
2880 ' Index_out(*)    ' BINARY BIT REVERSED OUTPUT VECTOR INDEX
2890 ' Vector_out(*)   ' BIT REVERSE SORTED OUTPUT VECTOR .
2900 !**********************************************
2910 FOR I=0 TO N_power     ' NULL BIT INDEX WORDS
```

```
2920   Index_in(I)=0
2930   Index_out(I)=0
2940 NEXT I
2950 FOR I=0 TO N_vector-1
2960   IF I<>0 THEN
2970      CALL Inc_binary(Index_in(*),N_power)
2980   END IF
2990   CALL Reflect(Index_in(*),N_power,Index_out(*))
3000   CALL Base_ten(Index_in(*),N_power,I_input)
3010   CALL Base_ten(Index_out(*),N_power,I_output)
3020   !**************************************
3030   !** BIT REVERSED INDICES OPERATION BELOW .**
3040   !****************************************
3050   Vector_out(I_output)=Vector_in(I_input)
3060 NEXT I
3070 SUBEND
3080 !*****************************************************************
3090 !********************** SUBROUTINE INC_BINARY ***********************
3100 !*****************************************************************
3110 !*        THIS SUBROUTINE PERFORMS A BINARY INCREMENT OPERATION ON THE *
3120 !* DEPOSITED BINARY VECTOR  ' Word_inc(*) '  AND RETURNS THE RESULT IN *
3130 !* THE SAME VARIABLE .                                                 *
3140 !*****************************************************************
3150 SUB Inc_binary(Word_inc(*),N_power)
3160 Carry_flag=0
3170 Done_flag=0
3180 I=0
3190 WHILE Done_flag=0
3200   IF I=0 THEN
3210      IF Word_inc(I)=0 THEN
3220         Word_inc(I)=1
3230         Done_flag=1
3240      ELSE
3250         Word_inc(I)=0
3260         Carry_flag=1
3270      END IF
3280   ELSE
3290      IF Carry_flag=1 THEN
3300         IF Word_inc(I)=0 THEN
3310            Word_inc(I)=1
3320            Done_flag=1
3330         ELSE
3340            Word_inc(I)=0
3350            Carry_flag=1
3360         END IF
3370      END IF
3380  END IF
3390  I=I+1
3400  IF I=N_power THEN
3410     Done_flag=1
3420  END IF
3430 END WHILE
3440 SUBEND
3450 !******************************************************************
3460 !*********************** SUBROUTINE REFLECT ************************
3470 !******************************************************************
3480 !*     THIS SUBROUTINE TRANSPOSES THE POSITION OF THE BITS IN THE INPUT *
3490 !* VECTOR  TO  OPPOSITE  POSITIONS  WITH RESPECT TO THE CENTROID OF THE *
3500 !* BINARY WORD .                                                        *
3510 !******************************************************************
3520 SUB Reflect(Word_in(*),N_power,Word_out(*))
3530 FOR I=0 TO N_power-1
3540  Word_out(I)=Word_in(N_power-I-1)
3550 NEXT I
3560 SUBEND
3570 !*****************************************************************
```

```
3580 !*********************** SUBROUTINE BASE_TEN ******************** ****
3590 !*******************************************************************
3600 !*       THIS SUBROUTINE CONVERTS THE DEPOSITED BINARY VECTOR TO A BASE *
3610 !* TEN INTEGER.THE BASE TEN NUMBER IS RETURNED IN THE VARIABLE 'X_out'.*
3620 !*******************************************************************
3630 SUB Base_ten(Word_in(*),N_power,X_out)
3640 X_out=0
3650 FOR I=0 TO N_power-1
3660  X_out=X_out+Word_in(I)*2^I
3670 NEXT I
3680 SUBEND
3690 !*******************************************************************
3700 !********************** SUBROUTINE BUTTERFLY ***********************
3710 !*******************************************************************
3720 !*       THIS SUBROUTINE GENERATES THE NECESSARY  INDICES  DEFINING THE *
3730 !* BUTTERFLIES WHICH  PERFORM  THE  IN-PLACE  COMPUTATIONS  OF  A FAST *
3740 !* FOURIER TRANSFORM .                                                 *
3750 !*******************************************************************
3760 SUB Butterfly(N_point,V_point,Stage,P(*),Q(*))
3770 !*************************************************************
3780 !************** DEFINITION OF LOCAL VARIABLES ****************
3790 !*************************************************************
3800 ! N_point      ! NUMBER OF POINTS IN FOURIER TRANSFORM .
3810 ! V_point      ! LOG BASE TWO OF NUMBER OF TRANSFORM POINTS.
3820 ! Stage        ! STAGE OF TRANSFORM VECTOR PROCESSING .
3830 ! Span         ! WIDTH OF ROW SPAN OF BUTTERFLY .
3840 ! N_butterfly  ! NUMBER OF BUTTERFLIES IN TRANSFORM STAGE.
3850 ! N_cross      ! NUMBER OF BUTTERFLIES FOUND .
3860 ! Up_cross     ! POSITION OF UPPER BUTTERFLY BRANCH.
3870 ! Low_cross    ! POSITION OF LOWER BUTTERFLY BRANCH .
3880 ! P(*)         ! 'P' INDEX OF BUTTERFLY 'N_cross' .
3890 ! Q(*)         ! 'Q' INDEX OF BUTTERFLY 'N_cross' .
3900 !*************************************************************
3910 Span=2^Stage
3920 !***************************
3930 !* DEFINE INITIAL BUTTERFLY *
3940 !***************************
3950 Up_cross=0
3960 Low_cross=Span
3970 N_cross=-1
3980 IF Span>1 THEN                      ! TEST OUT CASE OF STAGE ZERO
3990   WHILE N_cross<N_point/2-Span
4000      FOR I=Up_cross TO Low_cross-1
4010         N_cross=N_cross+1
4020         P(N_cross)=I
4030         Q(N_cross)=I+Span
4040      NEXT I
4050      Up_cross=Q(N_cross)+1
4060      Low_cross=Up_cross+Span
4070   END WHILE
4080 ELSE
4090   FOR I=0 TO N_point/2-1
4100      P(I)=2*I
4110      Q(I)=2*I+1
4120   NEXT I
4130 END IF
4140 SUBEND
     !*******************************************************************
     ************************* FUNCTION MODULO ***************************
     *******************************************************************
4... !*        THIS  FUNCTION   RETURNS    THE  MODULO VALUE  OF AN  INTEGER *
4190 !* ARGUMENT WRT THE MODULO LIMIT SPECIFIED AS ' Mod_max' .             *
4200 !*******************************************************************
4210 DEF FNModulo(Number,Mod_max)
4220 Dummy=INT(Number/Mod_max)
4230 N_mod=Number-Dummy*Mod_max
```

```
4240 RETURN N_mod
4250 FNEND
4260 !***********************************************************************
4270 !********************** SUBROUTINE PRODUCT_COMPLEX **********************
4280 !***********************************************************************
4290 !*  THIS SUBROUTINE PERFORMS A COMPLEX MULTIPLICATION OPERATION ON THE *
4300 !* DEPOSITED  ' X_real + X_image '   AND   ' Y_real + Y_image '  INPUT *
4310 !* VARIABLES    AND    RETURNS    THE    RESULT    IN    THE   VARIABLES *
4320 !* ' Z_real + Z_image ' .                                              *
4330 !***********************************************************************
4340 SUB Product_complex(X_real,X_image,Y_real,Y_image,Z_real,Z_image)
4350 !************************************************************
4360 !************** DEFINITION OF LOCAL VARIABLES **************
4370 !************************************************************
4380 ! X_real    ! REAL PART OF FIRST INPUT VARIABLE .
4390 ! X_image   ! IMAGINARY PART OF FIRST INPUT VARIABLE.
4400 ! Y_real    ! REAL PART OF SECOND INPUT VARIABLE .
4410 ! Y_image   ! IMAGINARY PART OF SECOND INPUT VARIABLE
4420 ! Z_real    ! REAL PART OF THE PRODUCT OF  INPUT VARIABLES .
4430 ! Z_image   ! IMAGINARY PART OF PRODUCT SUM OF INPUT VARIABLES
4440 !************************************************************
4450 Z_real=X_real*Y_real-(X_image*Y_image)
4460 Z_image=X_real*Y_image+X_image*Y_real
4470 SUBEND
4480 !***********************************************************************
4490 !************************* SUBROUTINE PHASOR *************************
4500 !***********************************************************************
4510 !*        THIS SUBROUTINE  COMPUTES  THE REAL AND IMAGINARY PARTS OF AN *
4520 !* EXPONENTIAL UNIT TRANSFORM PHASOR RAISED TO THE POWER ' R_power '.  *
4530 !***********************************************************************
4540 SUB Phasor(Pie,N,R,W_real,W_image)
4550 W_real=COS(2*Pie*R/N)
4560 W_image=SIN(2*Pie*R/N)
4570 SUBEND
4580 !***********************************************************************
4590 !*********************** SUBROUTINE ZERO_FILL ***********************
4600 !***********************************************************************
4610 !*          THIS SUBROUTINE EXTENDS THE LENGTH OF THE RECORD TO THE NEXT *
4620 !* HIGHEST POWER OF TWO BY FILLING THE REMAINDER WITH ZEROES .          *
4630 !***********************************************************************
4640 SUB Zero_fill(Dummy(*),N_in,N_out)
4650 V_in=INT(LOG(N_in)/LOG(2))          ! COMPUTE POWER OF TWO OF DATA RECORD.
4660 N_out=2^(V_in+1)                    ! INCREASE RECORD LENGTH TO NEXT HIGH-
4670 REDIM Dummy(N_out-1)                ! EST POWER OF TWO .
4680 FOR I=N_in TO N_out-1
4690   Dummy(I)=0                        ! ZERO FILL REMAINDER OF DATA RECORD .
4700 NEXT I
4710 SUBEND
4720 !***********************************************************************
4730 !*********************** SUBROUTINE INTEGRAND ***********************
4740 !***********************************************************************
4750 !*            THIS SUBROUTINE COMPUTES THE DIRECTIONALLY SIFTED TEMPORAL *
4760 !* RECORD  OF THE  SEA SURFACE  UNIT NORMAL ANGLES 'Phi' AND 'Theta' . *
4770 !* THAT IS , THE  FOURIER  TRANSFORM  IS  LATER  TAKEN  OF THE SINE OF *
4780 !* 'Phi' MODULATED  BY A  DIRECTIONAL GAUSSIAN SIFTER WITH 'Theta' AND *
4790 !* 'Gamma' AS  ARGUMENTS . REFER  TO  SECTION  ENTITLED 'Determination *
4800 !* of Wave Direction ' IN MAIN REPORT FOR RELATED THEORY.              *
4810 !***********************************************************************
4820 SUB Integrand(Phi(*),Theta(*),Pie,N_point,Gamma,Vector(*))
4830 RAD
4840 !************************************************************
4850 !************** DEFINTION OF LOCAL VARIABLES **************
4860 !************************************************************
4870 ! Phi(*)        ! ELEVATIONAL ANGLE OF UNIT NORMAL .
4880 ! Theta(*)      ! AZIMUTHAL ANGLE OF UNIT NORMAL .
4890 ! Beta          ! LINE OF ACTION ADJUSTED Theta(I) .
```

```
4900 ! N_point      ! LENGTH OF TEMPORAL DATA RECORD TO BE SIFTED.
4910 ! Gamma        ! CURRENT SIFTING DIRECTION AZIMUTH ANGLE .
4920 ! Sigma        ! STANDARD DEVIATION OF SIFTING OPERATOR .
4930 ! Vector(*)    ! RESULTING DIRECTIONAL FOURIER INTEGRAND .
4940 !*******************************************************
4950 FOR I=0 TO N_point-1
4960    Vector(I)=SIN(Phi(I))*COS(Theta(I)-Gamma)
4970 NEXT I
4980 SUBEND
4990 !*******************************************************************
5000 !*********************** SUBROUTINE FREQ_BASE ***********************
5010 !*******************************************************************
5020 !*        THIS  SUBROUTINE  COMPUTES THE FREQUENCY BASE VECTOR FOR THE *
5030 !* RESULTANT OF THE DIRECTIONAL FOURIER TRANSFORM .                   *
5040 !*******************************************************************
5050 SUB Freq_base(N_point,N_frequency,F_sample,Frequency(*))
5060 F_delta=F_sample/N_point
5070 FOR I=0 TO N_frequency-1
5080   Frequency(I)=I*F_delta
5090 NEXT I
5100 SUBEND
5110 !*******************************************************************
5120 !********************** SUBROUTINE STUFF_ARRAY **********************
5130 !*******************************************************************
5140 !*        THIS SUBROUTINE INSERTS THE DEPOSITED COLUMN VECTOR INTO THE *
5150 !* ARRAY SHOWN IN THE COMMON BLOCK 'FOURIER' .                        *
5160 !*******************************************************************
5170 SUB Stuff_array(Dummy(*),N_dummy,K_column)
5180 COM /Fourier/ Spectrum(512,36),Frequency(512),Direction(36)
5190 FOR I=0 TO N_dummy-1
5200    Spectrum(I,K_column)=Dummy(I)
5210 NEXT I
5220 SUBEND
5230 !*******************************************************************
5240 !********************** SUBROUTINE FIND_PEAKS ***********************
5250 !*******************************************************************
5260 !*        THIS  SUBROUTINE DETERMINES THE LOCATION OF THE DIRECTIONAL *
5270 !* PEAKS  IN  THE  SPECTRUM  FOR  EACH  FREQUENCY  COMPONENT  AND ALSO *
5280 !* DETERMINES THE RELATIVE CONTRIBUTION OF EACH FREQUENCY COMPONENT .  *
5290 !*******************************************************************
5300 SUB Find_peaks(Frequency(*),Direction(*),Bearing(*),Rel_cont(*),N_row,N_c
olumn)
5310 DIM Dummy(256)
5320 Pie=4*ATN(1)
5330 Sum_spec=0
5340 FOR I=0 TO N_row-1
5350    CALL Pull_vector(Dummy(*),N_column,I)
5360    CALL Peak_detect(Dummy(*),N_column,Maximum,I_maximum)
5370    Bearing(I)=Direction(I_maximum)*Pie/180
5380    Sum_spec=Sum_spec+Maximum
5390    Rel_cont(I)=Maximum
5400 NEXT I
5410 !**********************************
5420 !* SCALE RELATIVE CONTRIBUTION VECTOR *
5430 !**********************************
5440 FOR I=0 TO N_row-1
5450    Rel_cont(I)=Rel_cont(I)/Sum_spec
5460 NEXT I
5470 SUBEND
5480 !*******************************************************************
5490 !********************** SUBROUTINE PULL_VECTOR **********************
5500 !*******************************************************************
5510 !*        THIS  SUBROUTINE  PULLS  A ROW VECTOR FROM THE MATRIX IN THE *
5520 !* COMMON STATEMENT BELOW. THIS ROW VECTOR IS RETURNED IN THE VARIABLE *
5530 !* ' Dummy(*) ' .                                                     *
5540 !*******************************************************************
```

```
5550 SUB Pull_vector(Dummy(*),N_dummy,K_row)
5560 COM /Fourier/ Spectrum(512,36),Frequency(512),Direction(36)
5570 FOR J=0 TO N_dummy-1
5580    Dummy(J)=Spectrum(K_row,J)
5590 NEXT J
5600 SUBEND
5610 !************************************************************************
5620 !*********************** SUBROUTINE PEAK_DETECT *************************
5630 !************************************************************************
5640 !*          THIS  SUBROUTINE  FINDS  THE MAXIMUM ELEMENT IN THE VECTOR *
5650 !* ' Dummy(*) ' AND  RETURNS  IT  IN THE VARIABLE 'Maximum' ALONG WITH *
5660 !* ITS INDEX IN THE VARIABLE ' I_maximum ' .                          *
5670 !************************************************************************
5680 SUB Peak_detect(Dummy(*),N_dummy,Maximum,I_maximum)
5690 Maximum=0
5700 FOR I=0 TO N_dummy-1
5710    IF Dummy(I)>Maximum THEN
5720       Maximum=Dummy(I)
5730       I_maximum=I
5740    END IF
5750 NEXT I
5760 SUBEND
5770 !************************************************************************
5780 !************************ SUBROUTINE READFILE3 *************************
5790 !************************************************************************
5800 !*       THIS SUBROUTINE READS THREE DATA VECTORS FROM DISK STORAGE OF *
5810 !* EQUAL LENGTH AND BOOTS THEM INTO THE DUMMY VECTORS X(*),Y(*),Z(*).  *
5820 !************************************************************************
5830 SUB Readfile3(Name$,Job$,Medium$,N_data,X(*),Y(*),Z(*))
5840 DIM File_name$[40]
5850 !****************************************************************
5860 !******************* DEFINITION OF VARIABLES ******************
5870 !****************************************************************
5880 ! Name$         ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
5890 ! Job$          ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
5900 ! Medium$       ! ADDRESS OF MASS STORAGE MEDIUM
5910 ! N_data        ! NUMBER OF DATA ELEMENTS IN EACH VECTOR .
5920 !****************************************************************
5930 !**********************************
5940 ! ASSIGN BUFFER I/O PATH TO FILE *
5950 !**********************************
5960 IF Medium$=":INTERNAL" THEN
5970    File_name$=Name$&Medium$
5980 ELSE
5990    File_name$=Medium$&Name$
6000 END IF
6010 ASSIGN @Path_1 TO File_name$
6020 !**********************************
6030 !******* READ  JOB LABEL *********
6040 !**********************************
6050 ENTER @Path_1;Job$
6060 !**********************************
6070 !*** ENTER NUMBER OF ELEMENTS ****
6080 !**********************************
6090 ENTER @Path_1;N_data
6100 !**********************************
6110 !** CORRECTLY SIZE DATA VECTOR ***
6120 !**********************************
6130 REDIM X(N_data-1),Y(N_data-1),Z(N_data-1)
6140 !**********************************
6150 !******* READ  DATA ARRAY ********
6160 !**********************************
6170 ENTER @Path_1;X(*),Y(*),Z(*)
6180 !**********************************
6190 !***** CLOSE FILE AND BUFFER *****
6200 !**********************************
```

```
6210 ASSIGN @Path_1 TO *
6220 REDIM X(4096),Y(4096),Z(4096)
6230 SUBEND
6240 !**************************************************************
6250 !*********************** SUBROUTINE WRITEFILE3 ****************
6260 !**************************************************************
6270 !*      THIS SUBROUTINE ACCEPTS THREE DATA VECTORS OF EQUAL LENGTH AND *
6280 !* WRITES  THEM  TO  A  DISK STORAGE FILE UNDER THE FILENAME SPECIFIED *
6290 !* BY THE USER .                                                      *
6300 !**************************************************************
6310 SUB Writefile3(Name$,Job$,Medium$,N_data,X(*),Y(*),Z(*))
6320 DIM File_name$[40]
6330 !****************************************************
6340 !****************** DEFINITION OF VARIABLES ***********
6350 !****************************************************
6360 ! Name$         ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
6370 ! Job$          ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
6380 ! Medium$       ! ADDRESS OF MASS STORAGE MEDIUM
6390 ! N_data        ! NUMBER OF DATA ELEMENTS IN EACH VECTOR .
6400 !****************************************************
6410 !******************************
6420 !* CREATE DATA FILE FOR STORAGE **
6430 !******************************
6440 File_size=INT(N_data/9)
6450 IF Medium$=":INTERNAL" THEN
6460    File_name$=Name$&Medium$
6470 ELSE
6480    File_name$=Medium$&Name$
6490 END IF
6500 CREATE BDAT File_name$,File_size
6510 !********************************
6520 ! ASSIGN BUFFER I/O PATH TO FILE *
6530 !********************************
6540 ASSIGN @Path_1 TO File_name$
6550 !********************************
6560 !** CORRECTLY SIZE DATA VECTOR ***
6570 !********************************
6580 REDIM X(N_data-1),Y(N_data-1),Z(N_data-1)
6590 !********************************
6600 !******* STORE JOB LABEL *********
6610 !********************************
6620 OUTPUT @Path_1;Job$
6630 !********************************
6640 !**** STORE NUMBER OF ELEMENTS ***
6650 !********************************
6660 OUTPUT @Path_1;N_data
6670 !********************************
6680 !******* STORE DATA ARRAY ********
6690 !********************************
6700 OUTPUT @Path_1;X(*),Y(*),Z(*)
6710 !********************************
6720 !***** CLOSE FILE AND BUFFER *****
6730 !********************************
6740 ASSIGN @Path_1 TO *
6750 SUBEND
6760 !**************************************************************
6770 !*********************** SUBROUTINE PRINT_OUT ******************
6780 !**************************************************************
6790 !*           THIS  SUBROUTINE  PRINTS  OUT  THE  BEARING AND RELATIVE *
6800 !* CONTRIBUTION  FOR  EACH  FREQUENCY  COMPONENT  IN  THE DIRECTIONAL *
6810 !* SPECTRUM .                                                         *
6820 !**************************************************************
6830 SUB Print_out(Frequency(*),Bearing(*),Rel_cont(*),N_data)
6843 Pie=4*ATN(1)
6850 PRINTER IS 6
6860 PRINT CHR$(12)
```

```
 6870 PRINT "****************************************************************
************"
 6880 PRINT "********************** FREQUENCY BEARINGS AND CONTRIBUTIONS *****
************"
 6890 PRINT "****************************************************************
*************"
 6900 PRINT
 6910 PRINT
 6920 PRINT "Frequency              Bearing (degrees)        Relative Contributio
n (%) "
 6930 PRINT
 6940 FOR I=0 TO N_data-1
 6950    PRINT USING Format_1;Frequency(I),INT(Bearing(I)*1800/Pie)/10,INT(Rel_
cont(I)*1000)/10
 6960 Format_1:   IMAGE DD.DDD,16X,DDD.D,22X,DD.DD
 6970 NEXT I
 6980 PRINT CHR$(12)
 6990 PRINTER IS 1
 7000 SUBEND
 7010  !***************************************************************
 7020  !*********************** SUBROUTINE WINDOWER ***********************
 7030  !***************************************************************
 7040  !*          THIS SUBROUTINE PERFORMS A TRIANGULAR WINDOWING OPERATION *
 7050  !* ON THE SUPPLIED VECTOR . THIS IS A PRELUDE TO A FOURIER TRANSFORM *
 7060  !* OPERATION AND IS INTENDED TO REDUCE SPECTRAL SIDE LOBING .        *
 7070  !***************************************************************
 7080  SUB Windower(Vector(*),N_vector)
 7090  FOR I=0 TO N_vector-1
 7100     IF I<N_vector/2 THEN
 7110        Vector(I)=Vector(I)*(2*I/N_vector)
 7120     ELSE
 7130        Vector(I)=Vector(I)*(1-2*(I-N_vector/2)/N_vector)
 7140     END IF
 7150  NEXT I
 7160  SUBEND
```

```
31 Aug 1987         20:40:22

 1000 !**************************************************************
 1010 !************************* PROGRAM SEE_SPEC ************************
 1020 !**************************************************************
 1030 !*          THIS PROGRAM BOOTS IN A  FREQUENCY SPECTRUM AND PERMITS THE *
 1040 !* USER TO PLOT AND EXAMINE IT .                                         *
 1050 !**************************************************************
 1060 DIM Frequency(4096),Magnitude(4096),Phase(4096)
 1070 DIM Name$[16],Medium$[20],Job$[80]
 1080 !*****************************************************
 1090 !************** DEFINITION OF LOCAL VARIABLES **************
 1100 !*****************************************************
 1110 T_sample=1/60
 1120 Pie=4*ATN(1)
 1130 Medium$="BASIC/DATA_FILE/"
 1140 Penc=2
 1150 !*****************************************************
 1160 PRINT CHR$(12)
 1170 INPUT "Enter FILENAME of SOURCE Data File .....",Name$
 1180 INPUT "Enter FREQUENCY LIMIT on Spectrum (Hertz) ...",F_max
 1190 !****************************************
 1200 !******** COMPUTE TIME BASE VECTOR *********
 1210 !****************************************
 1220 CALL Readfile3(Name$,Job$,Medium$,N_point,Frequency(*),Magnitude(*),Phase
(*))
 1230 PRINT CHR$(12)
 1240 PRINT Job$
 1250 N_limit=INT(F_max/Frequency(1))
 1260 !****************************************
 1270 !* CONVERT DATA TO FOURIER TRANSFORM SCALE *
 1280 !****************************************
 1290 REDIM Frequency(N_limit-1),Magnitude(N_limit-1),Phase(N_limit-1)
 1300 FOR I=0 TO N_limit-1
 1310    Magnitude(I)=Magnitude(I)*T_sample
 1320 NEXT I
 1330 S_max=MAX(Magnitude(*))
 1340 CALL Plot_file(Frequency(*),Magnitude(*),N_limit,0,F_max,-S_max,S_max,Pen
c,"Y")
 1350 INPUT "Hit Return to CONTINUE ...",A$
 1360 Penc=Penc+1
 1370 CALL Plot_file(Frequency(*),Phase(*),N_limit,0,F_max,-2*Pie,2*Pie,Penc,"Y
")
 1380 PRINT
 1390 PRINT "Total Record Length is ";N_point;" Points....."
 1400 INPUT "HIT RETURN ...",A$
 1410 GRAPHICS OFF
 1420 PRINT CHR$(12)
 1430 END
 1440 !**************************************************************
 1450 !*********************** SUBROUTINE READFILE3 ***********************
 1460 !**************************************************************
 1470 !*         THIS SUBROUTINE READS THREE DATA VECTORS FROM DISK STORAGE OF *
 1480 !* EQUAL LENGTH AND BOOTS THEM INTO THE DUMMY VECTORS X(*),Y(*),Z(*).  *
 1490 !**************************************************************
 1500 SUB Readfile3(Name$,Job$,Medium$,N_data,X(*),Y(*),Z(*))
 1510 DIM File_name$[40]
 1520 !*********************************************************
 1530 !****************** DEFINITION OF VARIABLES ******************
 1540 !*********************************************************
 1550 ! Name$        ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
 1560 ! Job$         ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
 1570 ! Medium$      ! ADDRESS OF MASS STORAGE MEDIUM
 1580 ! N_data       ! NUMBER OF DATA ELEMENTS IN EACH VECTOR .
 1590 !*********************************************************
 1600 !********************************
```

```
1610 ! ASSIGN BUFFER I/O PATH TO FILE *
1620 !*********************************
1630 IF Medium$=":INTERNAL" THEN
1640    File_name$=Name$&Medium$
1650 ELSE
1660    File_name$=Medium$&Name$
1670 END IF
1680 ASSIGN @Path_1 TO File_name$
1690 !*********************************
1700 !******** READ  JOB LABEL ********
1710 !*********************************
1720 ENTER @Path_1;Job$
1730 !*********************************
1740 !*** ENTER NUMBER OF ELEMENTS ****
1750 !*********************************
1760 ENTER @Path_1;N_data
1770 !*********************************
1780 !** CORRECTLY SIZE DATA VECTOR ***
1790 !*********************************
1800 REDIM X(N_data-1),Y(N_data-1),Z(N_data-1)
1810 !*********************************
1820 !******** READ  DATA ARRAY ********
1830 !*********************************
1840 ENTER @Path_1;X(*),Y(*),Z(*)
1850 !*********************************
1860 !***** CLOSE FILE AND BUFFER *****
1870 !*********************************
1880 ASSIGN @Path_1 TO *
1890 SUBEND
1900 !*************************************************************************
1910 !*********************** SUBROUTINE PLOT_FILE ***************************
1920 !*************************************************************************
1930 !*      THIS SUBROUTINE ACCEPTS TWO DATA VECTORS AND PLOTS ONE  VERSUS *
1940 !* THE OTHER . THE  USER  NEED  ONLY  SUPPLY  THE  LIMITS OF THE GIVEN *
1950 !* VECTORS AND THE DESIRED PLOTTING COLOR . SCALING AND AXES ARE AUTO- *
1960 !* MATICALLY PROVIDED BY THIS SUBROUTINE .                              *
1970 !*************************************************************************
1980 SUB Plot_file(Xdata(*),Ydata(*),Nplot,Xmin,Xmax,Ymin,Ymax,Penc,New$)
1990 COM /Plot_block/ Xscale,Yscale,Xoffset,Yoffset
2000 !************************************************************
2010 !************** DEFINTITION OF LOCAL VARIABLES *************
2020 !************************************************************
2030 ! Xdata(*)        ! ABSCISSA DATA VECTOR TO BE PLOTTED .
2040 ! Ydata(*)        ! ORDINATE DATA VECTOR TO BE PLOTTED .
2050 ! Nplot           ! NUMBER OF DATA POINTS IN VECTORS .
2060 ! Xmin            ! SMALLEST ELEMENT IN Xdata(*) VECTOR .
2070 ! Xmax            ! LARGEST ELEMENT IN Xdata(*) VECTOR .
2080 ! Ymin            ! SMALLEST ELEMENT IN Ydata(*) VECTOR .
2090 ! Ymax            ! LARGEST ELEMENT IN Ydata(*) VECTOR .
2100 ! Penc            ! DESIRED COLOR CODE OF PLOTTING COLOR  .
2110 ! New$            ! ORDERS THE ROUTINE TO CLEAR THE GRAPHICS
2120 White=1          ! DEFINE THE COLOR CODE FOR WHITE
2130 A_color=White    ! SET AXIS COLOR WHITE
2140 Xleft=0          ! DEFINE LEFT OF SCREEN     (Plotter Units)
2150 Xrail=20         ! DEFINE X AXIS RAIL        (Plotter Units)
2160 Xcenter=64       ! X COORD CENTER SCREEN     (Plotter Units)
2170 Xright=128       ! DEFINE RIGHT SCREEN       (Plotter Units)
2180 Ybottom=0        ! DEFINE LOWER SCREEN       (Plotter Units)
2190 Yrail=16         ! DEFINE Y AXIS RAIL        (Plotter Units)
2200 Ycenter=48       ! Y COORDCENTER SCREEN      (Plotter Units)
2210 Ytop=96          ! DEFINE TOP OF SCREEN      (Plotter Units)
2220 ! X_denom        ! DENOMINATOR OF X PLOTTING SCALE FACTOR .
2230 ! Y_denom        ! DENOMINATOR OF Y PLOTTING  SCALE FACTOR
2240 !************************************************************
2250 !************************************************
2260 !** CLEAR AND INITIALIZE GRAPHICS IF SPECIFIED *
```

```
2270 !********************************************
2280 IF News="Y" THEN
2290   GINIT 1.5
2300   GRAPHICS ON
2310   PEN White
2320   VIEWPORT Xleft;Xright,Ybottom,Ytop
2330   FRAME
2340   !********************************
2350   !* DRAW PROPER AXES FOR PLOTTING *
2360   !********************************
2370   IF Xmin<0 THEN
2380      IF Ymin<0 THEN         !******************************
2390         Xoffset=Xcenter     !*  FOUR QUAD AXES DRAWN HERE *
2400         Yoffset=Ycenter     !******************************
2410         X_denom=Xmax
2420         Y_denom=Ymax
2430         CALL Axis_draw(Xleft,Yoffset,Xright,Yoffset,A_color,-Xmax,Xmax)
2440         CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,-Ymax,Ymax)
2450      ELSE                   !******************************
2460         Xoffset=Xcenter     !* +/- X TYPE AXIS DRAWN HERE *
2470         Yoffset=Yrail       !******************************
2480         X_denom=Xmax
2490         Y_denom=Ymax-Ymin
2500         CALL Axis_draw(Xleft,Yoffset,Xright,Yoffset,A_color,-Xmax,Xmax)
2510         CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,Ymin,Ymax)
2520      END IF
2530   ELSE
2540      IF Ymin<0 THEN        !******************************
2550         Xoffset=Xrail      !* +/- Y TYPE AXIS DRAWN HERE *
2560         Yoffset=Ycenter    !******************************
2570         X_denom=Xmax-Xmin
2580         Y_denom=Ymax-Ymin
2590         CALL Axis_draw(Xoffset,Yoffset,Xright,Yoffset,A_color,Xmin,Xmax)
2600         CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,-Ymax,Ymax)
2610         Yoffset=Ybottom
2620      ELSE                  !******************************
2630         Xoffset=Xrail      !* * ONLY X&Y AXES DRAWN HERE *
2640         Yoffset=Ybottom    !******************************
2650         X_denom=Xmax-Xmin
2660         Y_denom=Ymax-Ymin
2670         CALL Axis_draw(Xoffset,Yoffset,Xright,Yoffset,A_color,Xmin,Xmax)
2680         CALL Axis_draw(Xoffset,Yoffset,Xoffset,Ytop,A_color,Ymin,Ymax)
2690      END IF
2700   END IF
2710   Xscale=(Xright-Xoffset)/X_denom
2720   Yscale=(Ytop-Yoffset)/Y_denom
2730 END IF
2740 !******************************
2750 !* DATA VECTORS PLOTTED BELOW *
2760 !******************************
2770 PENUP
2780 CALL Scaler(Xdata(0),Ydata(0),Xmin,Xmax,Ymin,Ymax,X_plot,Y_plot)
2790 PEN Penc
2800 MOVE X_plot,Y_plot
2810 FOR I=0 TO Nplot-1
2820   CALL Scaler(Xdata(I),Ydata(I),Xmin,Xmax,Ymin,Ymax,X_plot,Y_plot)
2830   DRAW X_plot,Y_plot
2840 NEXT I
2850 SUBEND
2860 !***********************************************************************
2870 !********************** SUBROUTINE AXIS_DRAW ***************************
2880 !***********************************************************************
2890 !*       THIS SUBROUTINE DRAWS AN AXIS FROM THE STARTING COORDINATE TO *
2900 !* THE FINAL ONE . IT ALSO QUANTIFIES THE ORIGIN AND TERMINUS OF  SAID *
2910 !* AXIS .                                                              *
2920 !***********************************************************************
2930 SUB Axis_draw(Xstart,Ystart,Xfinal,Yfinal,Axis_color,A_min,A_max)
2940 Pie=4*ATN(1)
2950 Delta=5
2960 PENUP
```

```
2970 PEN Axis_color
2980 PENUP
2990 MOVE Xstart,Ystart
3000 DRAW Xfinal,Yfinal
3010 PENUP
3020 CSIZE 3.0,.5
3030 CALL Rounder(A_min,3,A0)
3040 CALL Rounder(A_max,3,A1)
3050 IF Xstart=Xfinal THEN
3060   CALL Labelit(Xstart-Delta,Ystart,Pie/2,Axis_color,VAL$(A0))
3070   CALL Labelit(Xfinal-Delta,Yfinal-2*Delta,Pie/2,Axis_color,VAL$(A1))
3080 ELSE
3090   CALL Labelit(Xstart,Ystart-Delta,0,Axis_color,VAL$(A0))
3100   CALL Labelit(Xfinal-2*Delta,Ystart-Delta,0,Axis_color,VAL$(A1))
3110 END IF
3120 SUBEND
3130 !**************************************************************************
3140 !*********************** SUBROUTINE LABELIT *******************************
3150 !**************************************************************************
3160 !* THIS SUBROUTINE SIMPLY ACCEPTS THE GIVEN LABEL AND PLACES IT WHERE   *
3170 !* IT IS SPECIFIED (ie X,Y LOCATION) AT THE GIVEN TILT ANGLE . THE PEN  *
3180 !* COLOR 'Penc' IS ALSO  PROVIDED  BY  THE  USER  . THIS SAVES A LOT OF *
3190 !* REPETITIVE CODE .                                                    *
3200 !**************************************************************************
3210 SUB Labelit(X,Y,Tilt,Penc,Strng$)
3220 PENUP
3230 MOVE X,Y
3240 PEN Penc
3250 LDIR Tilt
3260 LABEL Strng$
3270 PENUP
3280 SUBEND
3290 !*************************************************************************
3300 !************************* SUBROUTINE SCALER *****************************
3310 !*************************************************************************
3320 !*      THIS SUBROUTINE SCALES THE DATA PASSED TO IT FOR CRT PLOTTING  *
3330 !* PURPOSES .                                                          *
3340 !*************************************************************************
3350 SUB Scaler(X_data,Y_data,Xmin,Xmax,Ymin,Ymax,X_plot,Y_plot)
3360 COM /Plot_block/ Xscale,Yscale,Xoffset,Yoffset
3370 X_plot=Xscale*(X_data-Xmin)+Xoffset
3380 Y_plot=Yscale*(Y_data-Ymin)+Yoffset
3390 SUBEND
3400 !*************************************************************************
3410 !*********************** SUBROUTINE ROUNDER ******************************
3420 !*************************************************************************
3430 !*       THIS  SUBROUTINE  ACCEPTS  A  NUMBER  OF ANY SIZE OR SIGN AND *
3440 !* ROUNDS IT TO THE SPECIFIED NUMBER OF DIGITS .                       *
3450 !*************************************************************************
3460 SUB Rounder(X_input,N_digits,X_rounded)
3470 !***********************************************************
3480 !************ DEFINITION OF LOCAL VARIABLES *************
3490 !***********************************************************
3500 ! X_input       ! INPUT NUMBER TO BE ROUNDED
3510 ! X_dummy       ! DUMMY VARIABLE USED TO PROTECT X_input
3520 ! N_digits      ! NUMBER OF DIGITS DISPLAYED AFTER ROUNDING
3530 ! X_rounded     ! ROUNDED EQUIVALENT OF X_input
3540 ! Sign          ! NUMERICAL POLARITY OF ROUNDED NUMBER
3550 ! Magnitude     ! ORDER OF MAGNITUDE OF INPUT NUMBER
3560 ! Mantissa      ! MANTISSA OF NUMBER UNDER ROUNDING
3570 ! ARGUMENT      ! ABBREVIATED VERSION OF MANTISSA.
3580 !***********************************************************
3590 IF X_input<>0 THEN
3600   X_dummy=X_input
3610   Sign=SGN(X_dummy)
3620   X_dummy=ABS(X_dummy)
3630   Magnitude=INT(LGT(X_dummy))
3640   Mantissa=X_dummy/(10^Magnitude)
3650   Argument=INT(Mantissa*10^(N_digits-1))/10^(N_digits-1)
3660   X_rounded=Sign*Argument*10^Magnitude
3670 ELSE
3680   X_rounded=X_input
3690 END IF
3700 SUBEND
```

Appendix E

SEA SURFACE REPRODUCTION SOFTWARE

```
31 Aug 1987        20:42:10

 1000  !***************************************************************
 1010  !*********************** PROGRAM MAKE_MODEL ********************
 1020  !***************************************************************
 1030  !*             THIS PROGRAM GENERATES ALL OF THE  NECESSARY INFORMATION *
 1040  !* TO CREATE  A MODEL  OF THE  SEA SURFACE  BASED  ON  WAVE COMPUTER *
 1050  !* MEASUREMENTS .                                                *
 1060  !***************************************************************
 1070  COM /Wavelength/ Z0_mag(4096),Zdx_mag(4096),Zdy_mag(4096)
 1080  DIM Z0_sigh(4096),Zdx_sigh(4095),Zdy_sigh(4096),Dummy(4096)
 1090  DIM Name$[16],Name_ang$[16],Name_spec$[16],Name_mod$[16],Job$[80]
 1100  DIM Phi(4096),Theta(4096),Dz_dx(4096),Dz_dy(4096)
 1110  DIM Frequency(4096),Amplitude(4096),Phase(4096)
 1120  DIM Bearing(4096),Wavelength(4096),Velocity(4096)
 1130  PRINT CHR$(12)
 1140  !*********************************************************
 1150  !************** DEFINITION OF PROGRAM VARIABLES **************
 1160  !*********************************************************
 1170  ! Z0_mag(*)        ! Z VECTOR MAGNITUDE SPECTUM.       (Feet/Hertz)
 1180  ! Dz_dx(*)         ! PARTIAL DERIVATIVE OF Z WRT X.(Dimensionless)
 1190  ! Dz_dy(*)         ! PARTIAL DERIVATIVF OF Z WRT Y.(Dimensionless)
 1200  ! Phi(*)           ! ELEVATIONAL ANGLE OF UNIT NORMAL.   (Radians)
 1210  ! Theta(*)         ! AZIMUTHAL ANGLE OF UNIT NORMAL.     (Radians)
 1220  ! Dummy(*)         ! GENERAL PURPOSE DUMMY VARIABLE.
 1230  ! Frequency(*)     ! FRESUENCY OF WAVE COMPONENT.             (Hertz)
 1240  ! Zdx_mag(*)       ! dZ/dx  MAGNITUDE SPECTRUM .            (1/Hertz)
 1250  ! Zdy_mag(*)       ! dZ/dy  MAGNITUDE SPECTRUM.             (1/Hertz)
 1260  ! Zdx_sigh(*)      ! dZ/dx  PHASE SPECTRUM .               (Radians)
 1270  ! Zdy_sigh(*)      ! dZ/dy  PHASE SPECTRUM .               (Radians)
 1280  ! Amplitude(*)     ! AMPLITUDE OF WAVE COMPONENT.              (Feet)
 1290  ! Phase(*)         ! ANGULAR DELAY OF WAVE COMPONENT.     (Radians)
 1300  ! Bearing(*)       ! PLANE WAVE COMPONENT DIRECTION       (Radians)
 1310  ! Wavelength(*)    ! WAVELENGTH OF WAVE COMPONENT.             (Feet)
 1320  ! Velocity(*)      ! VELOCITY OF WAVE COMPONENT.      (Feet/second)
 1330  ! N_data           ! NUMBER OF DATA POINTS IN SPECTRUMS.
 1340  ! N_point          ! NUMBER OF FFT POINTS COMPUTED.
 1350  ! N_max            ! NUMBER OF MODEL FREQUENCY COMPONENTS CONSIDER
 1360  Medium$="BASIC/DATA_FILE/"
 1370  Pi=4*ATN(1)
 1380  T_sample=1/60
 1390  !*********************************************************
 1400  INPUT "Enter FILENAME of SOURCE DATA (Omit Extension) ....",Name$
 1410  INPUT "Enter COMPONENT TRUNCATION LENGTH ....",N_max
 1420  Name_z$=Name$&"_SPEC"
 1430  Name_dir$=Name$&"_DIR"
 1440  Name_ang$=Name$&"_ANG"
 1450  Name_sdx$=Name$&"_SDX"
 1460  Name_sdy$=Name$&"_SDY"
 1470  Name_mod$=Name$&"_MOD"
 1480  !*********************************************
 1490  !* BOOT IN DIRECTIONAL AND VERTICAL SPECTRAL DATA *
 1500  !*********************************************
 1510  DISP "************** BOOTING IN REQUIRED DATA FILES **************"
 1520  CALL Readfile3(Name_z$,Job$,Medium$,N_data,Frequency(*),Z0_mag(*),Z0_sig
h(*))
 1530  CALL Readfile3(Name_dir$,Job$,Medium$,N_data,Dummy(*),Bearing(*),Dummy(*
))
 1540  CALL Readfile3(Name_sdx$,Job$,Medium$,N_data,Dummy(*),Zdx_mag(*),Zdx_sig
h(*))
 1550  CALL Readfile3(Name_sdy$,Job$,Medium$,N_data,Dummy(*),Zdy_mag(*),Zdy_sig
h(*))
 1560  N_dummy=N_data
 1570  CALL Readfile3(Name_ang$,Job$,Medium$,N_dummy,Phi(*),Theta(*),Dummy(*))
 1580  DISP
 1590  N_point=2^INT(LOG(N_dummy)/LOG(2)+1)
```

```
1600  !*********************************************
1610  !********** EFFECT BEARING ADJUSTMENTS ************
1620  !*********************************************
1630  CALL Bearing_fix(Bearing(*),N_data,Pie)
1640  !*********************************************
1650  !************* STRAP-IN SPECTRAL PHASE *************
1660  !*********************************************
1670  MAT Phase=Z0_sigh
1680  !*********************************************
1690  !********** COMPUTE AMPLITUDE COEFFICIENTS *********
1700  !*********************************************
1710  CALL Amplitude(Z0_mag(*),N_da \,N_point,Amplitude(*))
1720  !*********************************************
1730  !**. COMPUTE WAVELENGTHS OF FREQUENCY COMPONENTS **
1740  !*********************************************
1750  CALL Wavelength(Bearing(*),N_data,Pie,Wavelength(*))
1760  !*********************************************
1770  !** COMPUTE WAVE FREQUENCY COMPONENT  VELOCITIES **
1780  !*********************************************
1790  CALL Velocity(Frequency(*),Wavelength(*),N_data,Velocity(*))
1800  !*********************************************
1810  !********** OUTPUT DATA TO DISK AND PRINTER ********
1820  !*********************************************
1830  INPUT "STORE Model Coefficients on DISK ? (Y/N) .....",A$
1840  IF A$="Y" THEN
1850     DISP "************ STORING SEA SURFACE MODEL COEFFICIENTS ************
"
 1860     CALL Writefile6(Name_mod$,Job$,Medium$,N_max,Frequency(*),Amplitude(*
),Phase(*),Bearing(*),Wavelength(*),Velocity(*))
 1870  END IF
 1880  INPUT "DUMP MODEL PARAMETERS to the PRINTER ? (Y/N)....",A$
 1890  IF A$="Y" THEN
 1900     DISP "***** OUTPUTTING SEA SURFACE MODEL COEFFICIENTS TO PRINTER ****
*"
 1910     CALL Print_out6(Frequency(*),Amplitude(*),Phase(*),Bearing(*),Wavelen
gth(*),Velocity(*),N_data)
 1920  END IF
 1930  CALL Video_game(1)
 1940  DISP "*************** MODEL RECORDING PROCESS COMPLETE ***************"
 1950  END
 1960 !****************************************************************
 1970 !************************** SUBROUTINE FFT **************************
 1980 !****************************************************************
 1990 !*          THIS  SUBROUTINE  PERFORMS  A  FAST FOURIER TRANSFORM ON THE *
 2000 !* DEPOSITED DATA VECTOR ' X_input(*) '. THE REAL PART OF THE SPECTRAL *
 2010 !* VECTOR  IS RETURNED IN THE VARIABLE ' F_real(*) ' AND THE IMAGINARY *
 2020 !* PART IS  RETURNED  IN  VARIABLE ' F_image(*) ' . IT IS IMPORTANT TO *
 2030 !* NOTE THAT , IN ORDER  FOR  THIS FFT ALGORITHM TO WORK THE NUMBER OF *
 2040 !* DATA POINTS UNDER ANALYSIS MUST BE A POWER OF TWO !!                *
 2050 !****************************************************************
 2060 SUB Fft(X_input(*),N_point,Pie,Magnitude(*),Phase(*))
 2070 DIM Real_1(4096),Image_1(4096),Real_2(4096),Image_2(4096)
 2080 DIM P_index(2048),Q_index(2048)
 2090 REDIM Real_1(N_point-1),Image_1(N_point-1)
 2100 REDIM Real_2(N_point-1),Image_2(N_point-1)
 2110 RAD
 2120 Pie=4*ATN(1)
 2130 V_point=INT(LOG(N_point)/LOG(2))
 2140 !*********************************************
 2150 !***** ORDER DATA VECTOR FOR INPUT OF TRANSFORM *****
 2160 !*********************************************
 2170 CALL Bit_reverse(X_input(*),N_point,V_point,Real_1(*))
 2180 !******************************
 2190 !* NULL IMAGINARY INPUT VECTOR *
 2200 !******************************
 2210 FOR I=0 TO N_point/2-1
```

```
2220   Image_1(I)=0
2230 NEXT I
2240 FOR I_stage=0 TO V_point-1             !  START STAGE STROBING LOOP
2250   CALL Butterfly(N_point,V_point,I_stage,P_index(*),Q_index(*))
2260   FOR J_butterfly=0 TO N_point/2-1     ! START BUTTERFLY STROBING LOOP .
2270      !**********************************
2280      !* DETERMINE BUTTERFLY BRANCH POINTS *
2290      !**********************************
2300      P=P_index(J_butterfly)
2310      Q=Q_index(J_butterfly)
2320      R_power=FNModulo(J_butterfly*2^(V_point-1-I_stage),N_point/2)
2330      CALL Phasor(Pie,N_point,R_power,W_real,W_image)
2340      CALL Product_complex(W_real,W_image,Real_1(Q),Image_1(Q),Dummy_real,
Dummy_image)
2350      !*********************************
2360      !* COMPUTE UPPER HALF OF BUTTERFLY *
2370      !*********************************
2380      Real_2(P)=Real_1(P)+Dummy_real
2390      Image_2(P)=Image_1(P)+Dummy_image
2400      !*********************************
2410      !* COMPUTE LOWER HALF OF BUTTERFLY *
2420      !*********************************
2430      Real_2(Q)=Real_1(P)-Dummy_real
2440      Image_2(Q)=Image_1(P)-Dummy_image
2450   NEXT J_butterfly
2460      !*********************************
2470      !* UPDATE NEXT CYCLE SOURCE VECTOR *
2480      !*********************************
2490      MAT Real_1=Real_2
2500      MAT Image_1=Image_2
2510 NEXT I_stage
2520 !********************************************
2530 !* DETERMINE MAGNITUDE AND PHASE OF SPECTRUM *
2540 !********************************************
2550 CALL Mag_phase(Real_2(*),Image_2(*),N_point,Magnitude(*),Phase(*))
2560 SUBEND
2570 !***************************************************************
2580 !************************* SUBROUTINE MAG_PHASE ***************
2590 !***************************************************************
2600 !*     THIS SUBROUTINE COMPUTES THE MAGNITUDE AND PHASE OF THE COMPLEX *
2610 !* VECTORS PROVIDE IN THE VARIABLES ' X_real(*) ' AND ' X_image(*) '. *
2620 !* THE RESULTING [illegible]NITUDE IS THEN STORED IN THE VECTOR  ' R_mag(*) ' *
2630 !* AND THE PHASE [illegible] STORED IN THE VECTOR ' P_phase(*) ' .          *
2640 !***************************************************************
2650 SUB Mag_phase(X_real(*),X_image(*),N_point,R_mag(*),P_phase(*))
2660 Pie=4*ATN(1)
2670 FOR I=0 TO N_point-1
2680   R_mag(I)=SQR(X_real(I)*X_real(I)+X_image(I)*X_image(I))
2690   IF X_real(I)<>0 THEN
2700      Phase=ATN(ABS(X_image(I)/X_real(I)))
2710   ELSE
2720      Phase=Pie/2
2730   END IF
2740   X_sign=SGN(X_real(I))
2750   Y_sign=SGN(X_image(I))
2760   IF Y_sign>=0 THEN
2770      IF X_sign>=0 THEN
2780         P_phase(I)=Phase
2790      ELSE
2800         P_phase(I)=Pie-Phase
2810      END IF
2820   ELSE
2830      IF X_sign>=0 THEN
2840         P_phase(I)=-Phase
2850      ELSE
2860         P_phase(I)=Phase-Pie
```

```
2870       END IF
2880    END IF
2890 NEXT I
2900 SUBEND
2910 !*****************************************************************
2920 !*********************** SUBROUTINE BIT_REVERSE *******************
2930 !*****************************************************************
2940 !*      THIS  SUBROUTINE  PERFORMS  A  BIT-REVERSAL  OPERATION ON THE *
2950 !* DEPOSITED INPUT VECTORS INDICES  . THIS IS  IN  PREPARATION  FOR AN *
2960 !* IN-PLACE FAST FOURIER TRANSFORM OPERATION .                     *
2970 !*****************************************************************
2980 SUB Bit_reverse(Vector_in(*),N_vector,N_power,Vector_out(*))
2990 DIM Index_in(16),Index_out(16)
3000 !*************************************************
3010 !************** DEFINITION OF LOCAL VARIABLES *************
3020 !*************************************************
3030 ! Vector_in(*)    ! INPUT VECTOR TO BE BIT REVERSE SORTED.
3040 ! N_power         ! LOG BASE TWO OF INPUT VECTOR LENGTH .
3050 ! N_vector        ! ACTUAL LENGTH OF INPUT VECTOR .
3060 ! Index_in(*)     ! BINARY INPUT VECTOR REFERENCE INDEX .
3070 ! Index_out(*)    ! BINARY BIT REVERSED OUTPUT VECTOR INDEX
3080 ! Vector_out(*)  ! BIT REVERSE SORTED OUTPUT VECTOR .
3090 !*************************************************
3100 FOR I=0 TO N_power     ! NULL BIT INDEX WORDS
3110   Index_in(I)=0
3120   Index_out(I)=0
3130 NEXT I
3140 FOR I=0 TO N_vector-1
3150   IF I<>0 THEN
3160      CALL Inc_binary(Index_in(*),N_power)
3170   END IF
3180   CALL Reflect(Index_in(*),N_power,Index_out(*))
3190   CALL Base_ten(Index_in(*),N_power,I_input)
3200   CALL Base_ten(Index_out(*),N_power,I_output)
3210   !*****************************************
3220   !** BIT REVERSED INDICES OPERATION BELOW .**
3230   !*****************************************
3240   Vector_out(I_output)=Vector_in(I_input)
3250 NEXT I
3260 SUBEND
3270 !*****************************************************************
3280 !*********************** SUBROUTINE INC_BINARY *********************
3290 !*****************************************************************
3300 !*         THIS SUBROUTINE PERFORMS A BINARY INCREMENT OPERATION ON THE *
3310 !* DEPOSITED BINARY VECTOR  ' Word_inc(*) '  AND RETURNS THE RESULT IN *
3320 !* THE SAME VARIABLE .                                              *
3330 !*****************************************************************
3340 SUB Inc_binary(Word_inc(*),N_power)
3350 Carry_flag=0
3360 Done_flag=0
3370 I=0
3380 WHILE Done_flag=0
3390   IF I=0 THEN
3400      IF Word_inc(I)=0 THEN
3410         Word_inc(I)=1
3420         Done_flag=1
3430      ELSE
3440         Word_inc(I)=0
3450         Carry_flag=1
3460      END IF
3470   ELSE
3480      IF Carry_flag=1 THEN
3490         IF Word_inc(I)=0 THEN
3500            Word_inc(I)=1
3510            Done_flag=1
3520         ELSE
```

```
3530           Word_inc(I)=0
3540           Carry_flag=1
3550        END IF
3560     END IF
3570  END IF
3580  I=I+1
3590  IF I=N_power THEN
3600     Done_flag=1
3610  END IF
3620 END WHILE
3630 SUBEND
3640 !************************************************************************
3650 !************************* SUBROUTINE REFLECT ***************************
3660 !************************************************************************
3670 !*     THIS SUBROUTINE TRANSPOSES THE POSITION OF THE BITS IN THE INPUT *
3680 !* VECTOR  TO  OPPOSITE  POSITIONS  WITH RESPECT TO THE CENTROID OF THE *
3690 !* BINARY WORD .                                                        *
3700 !************************************************************************
3710 SUB Reflect(Word_in(*),N_power,Word_out(*))
3720 FOR I=0 TO N_power-1
3730  Word_out(I)=Word_in(N_power-I-1)
3740 NEXT I
3750 SUBEND
3760 !***********************************************************************
3770 !************************ SUBROUTINE BASE_TEN **************************
3780 !***********************************************************************
3790 !*      THIS SUBROUTINE CONVERTS THE DEPOSITED BINARY VECTOR TO A BASE *
3800 !* TEN INTEGER.THE BASE TEN NUMBER IS RETURNED IN THE VARIABLE 'X_out'.*
3810 !***********************************************************************
3820 SUB Base_ten(Word_in(*),N_power,X_out)
3830 X_out=0
3840 FOR I=0 TO N_power-1
3850  X_out=X_out+Word_in(I)*2^I
3860 NEXT I
3870 SUBEND
3880 !************************************************************************
3890 !************************ SUBROUTINE BUTTERFLY **************************
3900 !***********************************************************************
3910 !*      THIS SUBROUTINE GENERATES THE NECESSARY  INDICES  DEFINING THE *
3920 !* BUTTERFLIES WHICH  PERFORM  THE  IN-PLACE  COMPUTATIONS  OF  A FAST *
3930 !* FOURIER TRANSFORM .                                                 *
3940 !***********************************************************************
3950 SUB Butterfly(N_point,V_point,Stage,P(*),Q(*))
3960 !*************************************************************
3970 !*************** DEFINITION OF LOCAL VARIABLES ***************
3980 !*************************************************************
3990 ! N_point      ! NUMBER OF POINTS IN FOURIER TRANSFORM .
4000 ! V_point      ! LOG BASE TWO OF NUMBER OF TRANSFORM POINTS.
4010 ! Stage        ! STAGE OF TRANSFORM VECTOR PROCESSING .
4020 ! Span         ! WIDTH OF ROW SPAN OF BUTTERFLY .
4030 ! N_butterfly  ! NUMBER OF BUTTERFLIES IN TRANSFORM STAGE.
4040 ! N_cross      ! NUMBER OF BUTTERFLIES FOUND .
4050 ! Up_cross     ! POSITION OF UPPER BUTTERFLY BRANCH.
4060 ! Low_cross    ! POSITION OF LOWER BUTTERFLY BRANCH .
4070 ! P(*)         ! 'P' INDEX OF BUTTERFLY 'N_cross' .
4080 ! Q(*)         ! 'Q' INDEX OF BUTTERFLY 'N_cross' .
4090 !*************************************************************
4100 Span=2^Stage
4110 !****************************
4120 !* DEFINE INITIAL BUTTERFLY *
4130 !****************************
4140 Up_cross=0
4150 Low_cross=Span
4160 N_cross=-1
4170 IF Span>1 THEN                    ! TEST OUT CASE OF STAGE ZERO
4180   WHILE N_cross<N_point/2-Span
```

```
4850 N_out=2^(V_in+1)                 ! INCREASE RECORD LENGTH TO NEXT HIGH-
4860 REDIM Dummy(N_out-1)             ! EST POWER OF TWO .
4870 FOR I=N_in TO N_out-1
4880   Dummy(I)=0                     ! ZERO FILL REMAINDER OF DATA RECORD .
4890 NEXT I
4900 SUBEND
4910 !****************************************************************
4920 !*********************** SUBROUTINE MAKE_SLOPES ******************
4930 !****************************************************************
4940 !*          THIS SUBROUTINE GENERATES THE SPATIAL DERIVATIVE VECTORS  *
4950 !* FROM THE ANGULAR FORMATTED WAVE COMPUTER DATA FILES .              *
4960 !****************************************************************
4970 SUB Make_slopes(Phi(*),Theta(*),N_data,Dz_dx(*),Dz_dy(*))
4980 FOR I=0 TO N_data-1
4990    Dz_dx(I)=-TAN(Phi(I))*COS(Theta(I))
5000    Dz_dy(I)=-TAN(Phi(I))*SIN(Theta(I))
5010 NEXT I
5020 SUBEND
5030 !****************************************************************
5040 !********************** SUBROUTINE BEARING_FIX *******************
5050 !****************************************************************
5060 !*       THIS SUBROUTINE ADJUSTS  ALL  FREQUENCY BEARING FIGURES SUCH *
5070 !* THAT THEY ALL EXIST IN QUADRANTS II AND III , THAT IS HEADING  FOR *
5080 !* SHORE .                                                           *
5090 !****************************************************************
5100 SUB Bearing_fix(Bearing(*),N_data,Pie)
5110 FOR I=0 TO N_data-1
5120    IF Bearing(I)<Pie/2 THEN
5130       Bearing(I)=Bearing(I)-Pie
5140    END IF
5150 NEXT I
5160 SUBEND
5170 !****************************************************************
5180 !*********************** SUBROUTINE WAVELENGTH *******************
5190 !****************************************************************
5200 !*        THIS  SUBROUTINE COMPUTES THE  WAVELENGTH  OF EACH SPECIFIC *
5210 !* FREQUENCY COMPONENT IN THE SPECTRUM .                              *
5220 !****************************************************************
5230 SUB Wavelength(Bearing(*),N_data,Pie,Wavelength(*))
5240 COM /Wavelength/ Z0_mag(4096),Zdx_mag(4096),Zdy_mag(4096)
5250 FOR I=0 TO N_data-1
5260    IF Zdy_mag(I)>Zdx_mag(I) THEN
5270       Wavelength(I)=2*Pie*ABS(SIN(Bearing(I)))*Z0_mag(I)/Zdy_mag(I)
5280    ELSE
5290       Wavelength(I)=2*Pie*ABS(COS(Bearing(I)))*Z0_mag(I)/Zdx_mag(I)
5300    END IF
5310 NEXT I
5320 SUBEND
5330 !****************************************************************
5340 !*********************** SUBROUTINE VELOCITIES *******************
5350 !****************************************************************
5360 !*           THIS  SUBROUTINE  COMPUTES  THE  WAVE  FRONT  VELOCITIES *
5370 !* FOR EACH FREQUENCY COMPONENT OF THE SEA SURFACE MODEL .            *
5380 !****************************************************************
5390 SUB Velocity(Frequency(*),Wavelength(*),N_data,Velocity(*))
5400 FOR I=0 TO N_data-1
5410    Velocity(I)=Wavelength(I)*Frequency(I)
5420 NEXT I
5430 SUBEND
5440 !*****************************************************************
5450 !*********************** SUBROUTINE AMPLITUDE *********************
5460 !*****************************************************************
5470 !*        THIS SUBROUTINE COMPUTES THE AMPLITUDE OF THE FOURIER SERIES *
5480 !* COEFFICIENTS FROM THE Z VECTOR SPECTRUM .                            *
5490 !*****************************************************************
5500 SUB Amplitude(Z0_mag(*),N_data,N_point,Amplitude(*))
```

```
5510 FOR I=0 TO N_data-1
5520    Amplitude(I)=Z0_mag(I)*2/N_point
5530 NEXT I
5540 SUBEND
5550 !*************************************************************
5560 !******************** SUBROUTINE WRITEFILE6 *******************
5570 !*************************************************************
5580 !*      THIS SUBROUTINE ACCEPTS THREE DATA VECTORS OF EQUAL LENGTH AND *
5590 !* WRITES  THEM  TO  A  DISK STORAGE FILE UNDER THE FILENAME SPECIFIED *
5600 !* BY THE USER .                                                       *
5610 !*************************************************************
5620 SUB Writefile6(Name$,Job$,Medium$,N_data,X1(*),X2(*),X3(*),X4(*),X5(*),X6
(*))
5630 DIM File_name$[40]
5640 !*******************************************************
5650 !***************** DEFINITION OF VARIABLES ***************
5660 !*******************************************************
5670 ! Name$        ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
5680 ! Job$         ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
5690 ! Medium$      ! ADDRESS OF MASS STORAGE MEDIUM
5700 ! N_data       ! NUMBER OF DATA ELEMENTS IN EACH VECTOR .
5710 !*******************************************************
5720 !********************************
5730 !* CREATE DATA FILE FOR STORAGE **
5740 !********************************
5750 File_size=INT(N_data/9)
5760 IF Medium$=":INTERNAL" THEN
5770    File_name$=Name$&Medium$
5780 ELSE
5790    File_name$=Medium$&Name$
5800 END IF
5810 CREATE BDAT File_name$,File_size
5820 !********************************
5830 ! ASSIGN BUFFER I/O PATH TO FILE *
5840 !********************************
5850 ASSIGN @Path_1 TO File_name$
5860 !********************************
5870 !** CORRECTLY SIZE DATA VECTOR ***
5880 !********************************
5890 REDIM X1(N_data-1),X2(N_data-1),X3(N_data-1)
5900 REDIM X4(N_data-1),X5(N_data-1),X6(N_data-1)
5910 !********************************
5920 !******* STORE JOB LABEL *********
5930 !********************************
5940 OUTPUT @Path_1;Job$
5950 !********************************
5960 !**** STORE NUMBER OF ELEMENTS ***
5970 !********************************
5980 OUTPUT @Path_1;N_data
5990 !********************************
6000 !******* STORE DATA ARRAY ********
6010 !********************************
6020 OUTPUT @Path_1;X1(*),X2(*),X3(*),X4(*),X5(*),X6(*)
6030 !********************************
6040 !***** CLOSE FILE AND BUFFER *****
6050 !********************************
6060 ASSIGN @Path_1 TO *
6070 SUBEND
6080 !*************************************************************
6090 !********************* SUBROUTINE READFILE3 ********************
6100 !*************************************************************
6110 !*        THIS SUBROUTINE READS THREE DATA VECTORS FROM DISK STORAGE OF *
6120 !* EQUAL LENGTH AND BOOTS THEM INTO THE DUMMY VECTORS X(*),Y(*),Z(*).  *
6130 !*************************************************************
6140 SUB Readfile3(Name$,Job$,Medium$,N_data,X(*),Y(*),Z(*))
6150 DIM File_name$[40]
```

```
6160 !***************************************************************
6170 !****************** DEFINITION OF VARIABLES *****************
6180 !***************************************************************
6190 ! Name$        ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
6200 ! Job$         ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
6210 ! Medium$      ! ADDRESS OF MASS STORAGE MEDIUM
6220 ! N_data       ! NUMBER OF DATA ELEMENTS IN EACH VECTOR .
6230 !***************************************************************
6240 !********************************
6250 ! ASSIGN BUFFER I/O PATH TO FILE *
6260 !********************************
6270 IF Medium$=":INTERNAL" THEN
6280    File_name$=Name$&Medium$
6290 ELSE
6300    File_name$=Medium$&Name$
6310 END IF
6320 ASSIGN @Path_1 TO File_name$
6330 !********************************
6340 !******** READ  JOB LABEL *********
6350 !********************************
6360 ENTER @Path_1;Job$
6370 !********************************
6380 !**** ENTER NUMBER OF ELEMENTS ****
6390 !********************************
6400 ENTER @Path_1;N_data
6410 !********************************
6420 !** CORRECTLY SIZE DATA VECTOR ***
6430 !********************************
6440 REDIM X(N_data-1),Y(N_data-1),Z(N_data-1)
6450 !********************************
6460 !******** READ  DATA ARRAY ********
6470 !********************************
6480 ENTER @Path_1;X(*),Y(*),Z(*)
6490 !********************************
6500 !****** CLOSE FILE AND BUFFER *****
6510 !********************************
6520 ASSIGN @Path_1 TO *
6530 SUBEND
6540 !**********************************************************************
6550 !*********************** SUBROUTINE PRINT_OUT6 ***********************
6560 !**********************************************************************
6570 !*        THIS  SUBROUTINE  PRINTS OUT A SIX VARIABLE DATA TABLE ON THE *
6580 !* LINE PRINTER .                                                      *
6590 !**********************************************************************
6600 SUB Print_out6(X1(*),X2(*),X3(*),X4(*),X5(*),X6(*),N_print)
6610 PRINTER IS 6
6620 Pie=4*ATN(1)
6630 PRINT CHR$(12)
6640 PRINT
6650 PRINT
6660 PRINT "**************************************************************
*************"
6670 PRINT "*********************** SEA SURFACE MODELING PARAMETERS **********
*************"
6680 PRINT "**************************************************************
*************"
6690 PRINT
6700 PRINT
6710 PRINT "Frequency   Amplitude    Phase    Bearing    Wavelength    Veloci
ty"
6720 PRINT
6730 FOR I=0 TO N_print-1
6740    PRINT USING Format_1;X1(I),X2(I),X3(I)*180/Pie,X4(I)*180/Pie,X5(I),X6(
I)
6750 Format_1:   IMAGE 1X,DD.DDD,5X,D.DDE,5X,SDDD.D,5X,SDDD.D,5X,D.DDE,5X,D.DD
E
```

```
6760 NEXT I
6770 PRINT CHR$(12)
6780 PRINTER IS 1
6790 SUBEND
6800 !*************************************************************************
6810 !********************** SUBROUTINE VIDEO_NOISE ***************************
6820 !*************************************************************************
6830 !*      .THIS SUBROUTINE GENERATES GENUINE VIDEO GAME SOUND EFFECTS FOR *
6840 !* MANY CYCLES AS YOU SPECIFY .                                          *
6850 !*************************************************************************
6860 SUB Video_game(N_cycles)
6870 IF N_cycles<1 THEN N_cycles=1
6880 FOR K=0 TO N_cycles-1
6890   FOR I=0 TO 4
6900      FOR J=0 TO 10
6910         BEEP J*100,I/100
6920      NEXT J
6930   NEXT I
6940 NEXT K
6950 SUBEND
```

31 Aug 1987   20:44:41

```
1000 !**********************************************************************
1010 !*********************** PROGRAM MAKE_WAVES **************************
1020 !**********************************************************************
1030 !*      THIS  PROGRAM READS IN A SEA SURFACE MODEL FROM THE GIVEN DATA *
1040 !* FILE RECONSTRUCTS IT AND PLOTS IT ON THE CRT .                      *
1050 !**********************************************************************
1060 COM /Labels/ Xlabel$[40],Ylabel$[40],Job$[80]
1070 COM /Waves/ Freq(512),Amp(512),Phase(512),Bear(512),Lamda(512),Speed(512)
1080 COM /Colors/ Data_color,Axis_color,Label_color
1090 COM /Answers/ X_ord(200),Y_ord(200),Z_coord(200,200)
1100 DIM File_in$[40],File_out$[40]
1110 RAD
1120 !****************************************************
1130 !************ DEFINITIONS OF PROGRAM VARIABLES **********
1140 !****************************************************
1150 Pie=4*ATN(1)
1160 ! Amp(*)            ! WAVE MODEL PEAK AMPLITUDE .
1170 ! Phase(*)          ! WAVE MODEL TEMPORAL PHASE .
1180 ! Bear(*)           ! WAVE MODEL WAVE DIRECTION .
1190 ! Lamda(*)          ! WAVE MODEL COMPONENT WAVELENGTH .
1200 ! Speed(*)          ! WAVE MODEL COMPONENT VELOCITY .
1210 ! X_ord(*)          ! X SPATIAL COORDINATE OF SEA SURFACE.
1220 ! Y_ord(*)          ! Y SPATIAL COORDINATE OF SEA SURFACE.
1230 ! N_waves           ! NUMBER OF WAVE COMPONENTS IN MODEL.
1240 Cross$="N"         ! DISABLE Y ORDINATE CROSS CONTOURS.
1250 Red=2              ! DEFINE RED PEN COLOR.
1260 Green=4            ! DEFINE GREEN PEN COLOR.
1270 Yellow=3           ! DEFINE YELLOW PEN COLOR.
1280 Aqua=5             ! DEFINE AQUA PEN COLOR .
1290 Blue=6             ! DEFINE BLUE PEN COLOR.
1300 White=1            ! DEFINE WHITE PEN COLOR.
1310 Data_color=Aqua    ! DEFINE DATA SURFACE COLOR.
1320 Label_color=Green  ! DEFINE LABEL COLOR.
1330 Axis_color=White   ! DEFINE AXIS COLOR.
1340 Medium_in$="BASIC/DATA_FILE/"
1350 Medium_out$="BASIC/WAVE_FILE/"
1360 !**************************************************
1370 PRINT CHR$(12)
1380 INPUT "Enter SPATIAL EXTENT of Simulation (Feet)...",L_max
1390 X_max=L_max
1400 Y_max=L_max
1410 Xlabel$="<- East "&VAL$(L_max)&" Feet"
1420 Ylabel$="-> North "&VAL$(L_max)&" Feet"
1430 INPUT "Enter DIFFERENTIAL SPATIAL STEP SIZE (Feet)...",Delta_x
1440 INPUT "Enter TEMPORAL FREQUENCY LIMIT (Hertz) ...",F_max
1450 Delta_y=Delta_x
1460 N_point=INT(L_max/Delta_x)
1470 !************************************
1480 !* DECIDE ON CROSS HATCHING SURFACE *
1490 !************************************
1500 IF N_point<40 THEN
1510    Cross$="Y"
1520 ELSE
1530    Cross$="N"
1540 END IF
1550 !****************************************************************
1560 !******** SIZE DOWN ORDINATE VECTORS AND COORDINATE ARRAYS ********
1570 !****************************************************************
1580 REDIM X_ord(N_point-1),Y_ord(N_point-1),Z_coord(N_point-1,N_point-1)
1590 INPUT "Enter FILENAME of Sea Surface Model (Omit Extension)...",File$
1600 File_in$=File$&"_MOD"
1610 CALL Readfile6(File_in$,Job$,Medium_in$,N_wave,Freq(*),Amp(*),Phase(*),Be
ar(*),Lamda(*),Speed(*))
1620 !**************************************
```

```
1630 !* COMPUTE TRUNCATED COMPONENT LENGTH *
1640 !*************************************
1650 N_freq=INT(F_max/Freq(1))
1660 !*************************************
1670 !**** DETERMINE SIMULATION OFFSET  ****
1680 !*************************************
1690 Max_wave=MAX(Amp(*))
1700 FOR I=0 TO N_point-1
1710   X_ord(I)=I*Delta_x
1720      BEEP
1730      DISP "*** OPERATION   ";INT(100*I/N_point);" PERCENT COMPLETE !! **"
1740   FOR J=0 TO N_point-1
1750      Y_ord(J)=J*Delta_y
1760      CALL Make_wave(X_ord(I),Y_ord(J),0,N_freq,Pie,Z_wave)
1770      Z_coord(I,J)=Z_wave
1780   NEXT J
1790 NEXT I
1800 Tilt=30
1810 !**************************************************
1820 !********** OFFER USER VIEWING OF SEA SURFACE **********
1830 !**************************************************
1840 WHILE Tilt<>0
1850   INPUT "Enter TILT ANGLE for 3-D Plot (Degrees)...[Enter 0 to ESCAPE.]",
Tilt
1860   IF Tilt<>0 THEN
1870      CALL Plot_3d(Tilt*Pie/360,N_point,Cross$)
1880   END IF
1890 END WHILE
1900 !*******************************************
1910 !********* OFFER DISK STORAGE OPTION *********
1920 !*******************************************
1930 INPUT "STORE Sea Surface on DISK ? (Y/N)...",A$
1940 IF A$="Y" THEN
1950    File_out$="SEA"&File$[5,7]&VAL$(F_max)&"."&VAL$(L_max)
1960    CALL Store_array(File_out$,Medium_out$,N_point)
1970 END IF
1980 GRAPHICS OFF
1990 CALL Video_game(1)
2000 DISP "******************** OPERATION COMPLETE !!! ********************"
2010 END
2020 !*****************************************************************
2030 !********************** SUBROUTINE MAKE_WAVE *********************
2040 !*****************************************************************
2050 !*        THIS  SUBROUTINE  GENERATES  A  SINGLE  VERTICAL  POINT ON A *
2060 !* RECONSTRUCTED SEA SURFACE FOR THE GIVEN 'X' AND 'Y' COORDINATE  AND *
2070 !* A POINT IN TIME 'T' .                                              *
2080 !*****************************************************************
2090 SUB Make_wave(X,Y,T,N_waves,Pie,Z_wave)
2100 COM /Waves/ Freq(512),Amp(512),Phase(512),Bear(512),Lamda(512),Speed(512)
2110 RAD
2120 Z_wave=0
2130 FOR I=0 TO N_waves-1
2140   Dot_prod=X*COS(Bear(I))+Y*SIN(Bear(I))
2150   Z_wave=Z_wave+Amp(I)*COS(2*Pie*(Dot_prod-Speed(I)*T)/Lamda(I)-Phase(I))
2160 NEXT I
2170 SUBEND
2180 !*****************************************************************
2190 !*********************** SUBROUTINE PLOT_3D **********************
2200 !*****************************************************************
2210 !* THIS SUBROUTINE IS  RESPONSIBLE  FOR  PLOTTING  THE  DATA  SURFACE  *
2220 !* GENERATED  BY  THE  MAIN  PROGRAM  MAIN FRAME . IT PERMITS FOR THE  *
2230 !* OPTIONS OF PLATE SIZE , AXIS TILT ANGLE  AXIS  COLOR  AND  SURFACE  *
2240 !* COLOR .                                                            *
2250 !*****************************************************************
2260 SUB Plot_3d(Thetax,Ngrid,Cross$)
2270 COM /Labels/ Xlabel$[40],Ylabel$[40],Job$[80]
```

```
2280 COM /Colors/ Data_color,Axis_color,Label_color
2290 COM /Answers/ X_ord(200),Y_ord(200),Z_coord(200,200)
2300 COM /Plotlink/ Xorigin,Yorigin,Left,Right,Bottom,Top
2310 COM /Scaler/ X_min,X_max,Y_min,Y_max,Z_max
2320 DIM Wave_max$[80]
2330 Pie=4*ATN(1)
2340 RAD
2350 !**************************************************
2360 !******** ESTABLISH DEFAULT PLOTTING PARAMETERS *******
2370 !**************************************************
2380 Left=0
2390 Right=125
2400 Bottom=0
2410 Top=125
2420 Slope=TAN(Thetax)
2430 Yorigin=(Right-Left)*Slope
2440 Xorigin=(Right-Left)/2
2450 Deltatext=(Top-Bottom)/25
2460 D_theta=Pie/36
2470 Offset=5
2480 X_min=MIN(X_ord(*))
2490 X_max=MAX(X_ord(*))
2500 Y_min=MIN(Y_ord(*))
2510 Y_max=MAX(Y_ord(*))
2520 Peak_max=MAX(Z_coord(*))
2530 Trough_max=MIN(Z_coord(*))
2540 Wave_max=(Peak_max-Trough_max)
2550 Wave_max$="Maximum Peak to Trough Depth is "&VAL$(INT(10*Wave_max)/10)&"
Feet"
2560 Z_off=3*Wave_max
2570 Z_max=5*Wave_max
2580 Tick=Z_max/50
2590 !****************************************
2600 !************* INITIALIZE PLOTTER ************
2610 !****************************************
2620 GINIT 1.25
2630 VIEWPORT Left,Right,Bottom,Top
2640 WINDOW Left,Right,Bottom,Top
2650 GRAPHICS ON
2660 PEN Frame_color
2670 FRAME
2680 CSIZE 2.5,.75
2690 PENUP
2700 !**********************************
2710 !************ DRAW AXES ************
2720 !**********************************
2730 PEN Axis_color
2740 MOVE Xorigin,Bottom
2750 DRAW Right,Xorigin*Slope
2760 DRAW Xorigin,Yorigin
2770 DRAW Left,Yorigin-Xorigin*Slope
2780 DRAW Xorigin,Bottom
2790 PENUP
2800 MOVE Xorigin,Yorigin
2810 PEN Axis_color
2820 DRAW Xorigin,Top
2830 PENUP
2840 PEN 0
2850 !**********************************************
2860 !************* LABEL ORDINATE AXES ***************
2870 !**********************************************
2880 Xtext=Xorigin/2+Deltatext
2890 Ytext=Bottom+Yorigin/4
2900 CALL Labelit(Xtext,Ytext,-Thetax+D_theta,Label_color,Xlabel$)
2910 Ytext=Bottom+Deltatext
2920 Xtext=Xorigin+Deltatext
```

```
2930 CALL Labelit(Xtext,Ytext,Thetax-D_theta,Label_color,Ylabel$)
2940 !**************************************************
2950 !****************** PLOT SURFACE DATA ******************
2960 !**************************************************
2970 !*******************************************
2980 !******** PLOT Y-ORDINATE CONTOUR LINES ********
2990 !*******************************************
3000 PENUP
3010 FOR I=0 TO Ngrid-1
3020   PEN Data_color
3030   FOR J=0 TO Ngrid-1
3040      CALL Scaler(X_ord(I),Y_ord(J),Z_coord(I,J)+Z_off,Slope,Xplot,Yplot)
3050      PLOT Xplot,Yplot
3060   NEXT J
3070   PENUP
3080 NEXT I
3090 PENUP
3100 !*******************************************
3110 !******* PLOT X ORDINATE CONTOUR LINES *********
3120 !*******************************************
3130 IF Cross$="Y" THEN
3140    FOR J=0 TO Ngrid-1
3150      PEN Data_color
3160      FOR I=0 TO Ngrid-1
3170         CALL Scaler(X_ord(I),Y_ord(J),Z_coord(I,J)+Z_off,Slope,Xplot,Yplo
t)
3180         PLOT Xplot,Yplot
3190      NEXT I
3200      PENUP
3210    NEXT J
3220 END IF
3230 !****************************************
3240 !******* ENTITLE PLOT OF SEA SURFACE ********
3250 !****************************************
3260 CSIZE 2.3,.75
3270 CALL Labelit(Left+5,Top-5,0,Label_color,Job$)
3280 CALL Labelit(Left+30,Top-8,0,2,Wave_max$)
3290 INPUT "Hit RETURN to CONTINUE ....",A$
3300 GRAPHICS OFF
3310 SUBEND
3320 !*************************************************************
3330 !*********************** SUBROUTINE SCALER *************************
3340 !*************************************************************
3350 !* THIS SUBROUTINE IS RESPONSIBLE FOR CONVERTING THE THREE DIMENSIONAL *
3360 !* DATA POINTS 'X' , 'Y' AND 'Z' INTO THE TWO DIMENSIONAL DATA  POINTS *
3370 !* 'Xplot' AND 'Yplot'.                                              *
3380 !*************************************************************
3390 SUB Scaler(X,Y,Z,Slope,Xplot,Yplot)
3400 COM /Plotlink/ Xorigin,Yorigin,Left,Right,Bottom,Top
3410 COM /Scaler/ X_min,X_max,Y_min,Y_max,Z_max
3420 Xplot=Xorigin*(1-(X-X_min)/(X_max-X_min)+(Y-Y_min)/(Y_max-Y_min))
3430 Yplot=Yorigin+(Top-(Right-Left)*Slope)*(Z/Z_max)-Xorigin*Slope*((X-X_min)
/(X_max-X_min)+(Y-Y_min)/(Y_max-Y_min))
3440 SUBEND
3450 !**************************************************************
3460 !********************** SUBROUTINE LABELIT ***********************
3470 !**************************************************************
3480 !* THIS SUBROUTINE SIMPLY ACCEPTS THE GIVEN LABEL AND PLACES IT WHERE   *
3490 !* IT IS SPECIFIED (ie X,Y LOCATION) AT THE GIVEN TILT ANGLE . THE PEN  *
3500 !* COLOR 'Penc' IS ALSO  PROVIDED  BY  THE  USER . THIS  SAVES A LOT OF *
3510 !* REPETITIVE CODE .                                                    *
3520 !**************************************************************
3530 SUB Labelit(X,Y,Tilt,Penc,Strng$)
3540 PEN Up
3550 MOVE X,Y
3560 PEN Penc
```

```
3570 LDIR Tilt
3580 LABEL Strng$
3590 PEN Up
3600 SUBEND
3610 !**********************************************************************
3620 !************************ SUBROUTINE READFILE6 ************************
3630 !**********************************************************************
3640 !*      THIS SUBROUTINE READS SIX DATA VECTORS FROM A DATA FILE AND *
3650 !* BOOTS THEM BACK TO THE MAINFRAME PROGRAM .                        *
3660 !**********************************************************************
3670 SUB Readfile6(Name$,Job$,Medium$,N_data,X1(*),X2(*),X3(*),X4(*),X5(*),X6(
*))
3680 DIM File_name$[40]
3690 !***************************************************************
3700 !****************** DEFINITION OF VARIABLES ******************
3710 !***************************************************************
3720 ! Name$         ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
3730 ! Job$          ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
3740 ! Medium$       ! ADDRESS OF MASS STORAGE MEDIUM
3750 ! N_data        ! NUMBER OF DATA ELEMENTS IN EACH VECTOR .
3760 !***************************************************************
3770 !*********************************
3780 !** ESTABLISH COMPLETE FILENAME **
3790 !*********************************
3800 IF Medium$=":INTERNAL" THEN
3810    File_name$=Name$&Medium$
3820 ELSE
3830    File_name$=Medium$&Name$
3840 END IF
3850 !*********************************
3860 ! ASSIGN BUFFER I/O PATH TO FILE *
3870 !*********************************
3880 ASSIGN @Path_1 TO File_name$
3890 !*********************************
3900 !******* READ  JOB LABEL ********
3910 !*********************************
3920 ENTER @Path_1;Job$
3930 !*********************************
3940 !**** READ  NUMBER OF ELEMENTS ***
3950 !*********************************
3960 ENTER @Path_1;N_data
3970 !*********************************
3980 !** CORRECTLY SIZE DATA VECTOR ***
3990 !*********************************
4000 REDIM X1(N_data-1),X2(N_data-1),X3(N_data-1)
4010 REDIM X4(N_data-1),X5(N_data-1),X6(N_data-1)
4020 !*********************************
4030 !******* READ DATA VECTORS *******
4040 !*********************************
4050 ENTER @Path_1;X1(*),X2(*),X3(*),X4(*),X5(*),X6(*)
4060 !*********************************
4070 !***** CLOSE FILE AND BUFFER *****
4080 !*********************************
4090 ASSIGN @Path_1 TO *
4100 SUBEND
4110 !**********************************************************************
4120 !*********************** SUBROUTINE STORE_ARRAY **********************
4130 !**********************************************************************
4140 !*        THIS SUBROUTINE STORES THE SEA SURFACE ARRAY ALONG WITH ITS *
4150 !* TWO ORDINATE VECTORS .                                             *
4160 !**********************************************************************
4170 SUB Store_array(File_name$,Medium$,N_data)
4180 COM /Answers/ X_ord(200),Y_ord(200),Z_coord(200,200)
4190 COM /Labels/ Xlabel$[40],Ylabel$[40],Job$[80]
4200 !*******************************************
4210 !********* DETERMINE COMPLETE FILENAME ********
```

```
4220 !**********************************************
4230 IF Medium$=":INTERNAL" THEN
4240    File_name$=File_name$&Medium$
4250 ELSE
4260    File_name$=Medium$&File_name$
4270 END IF
4280 File_size=INT((N_data^2+2*N_data)/9)
4290 !*******************************
4300 !* CREATE DATA FILE FOR STORAGE *
4310 !*******************************
4320 CREATE BDAT File_name$,File_size
4330 !*******************************
4340 !*** OPEN CHANNEL TO I/O PATH ***
4350 !*******************************
4360 ASSIGN @Path_1 TO File_name$
4370 !*******************************
4380 !* STORE VECTOR AND ARRAY SIZE **
4390 !*******************************
4400 OUTPUT @Path_1;N_data
4410 !*******************************
4420 !******* STORE JOB LABELS *******
4430 !*******************************
4440 OUTPUT @Path_1;Job$
4450 OUTPUT @Path_1;Xlabel$,Ylabel$
4460 !*******************************
4470 !***  STORE ORDINATE VECTORS  ***
4480 !*******************************
4490 REDIM X_ord(N_data-1),Y_ord(N_data-1)
4500 OUTPUT @Path_1;X_ord(*),Y_ord(*)
4510 !*******************************
4520 !*** STORE Z COORDINATE ARRAY ***
4530 !*******************************
4540 FOR I=0 TO N_data-1
4550    FOR J=0 TO N_data-1
4560       OUTPUT @Path_1;Z_coord(I,J)
4570    NEXT J
4580 NEXT I
4590 !*******************************
4600 !** CLOSE CHANNEL TO DATA FILE **
4610 !*******************************
4620 ASSIGN @Path_1 TO *
4630 SUBEND
4640 !**********************************************************************
4650 !*********************** SUBROUTINE VIDEO_NOISE ***********************
4660 !**********************************************************************
4670 !*      THIS SUBROUTINE GENERATES GENUINE VIDEO GAME SOUND EFFECTS FOR *
4680 !* MANY CYCLES AS YOU SPECIFY .                                        *
4690 !**********************************************************************
4700 SUB Video_game(N_cycles)
4710 IF N_cycles<1 THEN N_cycles=1
4720 FOR K=0 TO N_cycles-1
4730   FOR I=0 TO 4
4740      FOR J=0 TO 10
4750         BEEP J*100,1/100
4760      NEXT J
4770   NEXT I
4780 NEXT K
4790 SUBEND
```

```
31 Aug 1987          20:45:47

 1000 !****************************************************************
 1010 !************************ PROGRAM SEE_SPEC ************************
 1020 !****************************************************************
 1030 !*          THIS PROGRAM BOOTS IN A  FREQUENCY SPECTRUM AND PERMITS THE *
 1040 !* USER TO PLOT AND EXAMINE IT .                                    *
 1050 !****************************************************************
 1060 DIM Frequency(4096),Magnitude(4096),Phase(4096)
 1070 DIM Name$[16],Medium$[20],Job$[80]
 1080 !******************************************************
 1090 !************** DEFINITION OF LOCAL VARIABLES **************
 1100 !******************************************************
 1110 T_sample=1/60
 1120 Pie=4*ATN(1)
 1130 Medium$="BASIC/DATA_FILE/"
 1140 Penc=2
 1150 !******************************************************
 1160 PRINT CHR$(12)
 1170 INPUT "Enter FILENAME of SOURCE Data File .....",Name$
 1180 INPUT "Enter FREQUENCY LIMIT on Spectrum (Hertz) ...",F_max
 1190 !*****************************************
 1200 !********* COMPUTE TIME BASE VECTOR *********
 1210 !*****************************************
 1220 CALL Readfile3(Name$,Job$,Medium$,N_point,Frequency(*),Magnitude(*),Phase
(*))
 1230 PRINT CHR$(12)
 1240 PRINT Job$
 1250 N_limit=INT(F_max/Frequency(1))
 1260 !*****************************************
 1270 !* CONVERT DATA TO FOURIER TRANSFORM SCALE *
 1280 !*****************************************
 1290 REDIM Frequency(N_limit-1),Magnitude(N_limit-1),Phase(N_limit-1)
 1300 FOR I=0 TO N_limit-1
 1310    Magnitude(I)=Magnitude(I)*T_sample
 1320 NEXT I
 1330 S_max=MAX(Magnitude(*))
 1340 CALL Plot_file(Frequency(*),Magnitude(*),N_limit,0,F_max,-S_max,S_max,Pen
c,"Y")
 1350 INPUT "Hit Return to CONTINUE ...",A$
 1360 Penc=Penc+1
 1370 CALL Plot_file(Frequency(*),Phase(*),N_limit,0,F_max,-2*Pie,2*Pie,Penc,"Y
")
 1380 PRINT
 1390 PRINT "Total Record Length is ";N_point;" Points....."
 1400 INPUT "HIT RETURN ...",A$
 1410 GRAPHICS OFF
 1420 PRINT CHR$(12)
 1430 END
 1440 !****************************************************************
 1450 !********************** SUBROUTINE READFILE3 **********************
 1460 !****************************************************************
 1470 !*         THIS SUBROUTINE READS THREE DATA VECTORS FROM DISK STORAGE OF *
 1480 !* EQUAL LENGTH AND BOOTS THEM INTO THE DUMMY VECTORS X(*),Y(*),Z(*).  *
 1490 !****************************************************************
 1500 SUB Readfile3(Name$,Job$,Medium$,N_data,X(*),Y(*),Z(*))
 1510 DIM File_name$[40]
 1520 !*********************************************************
 1530 !****************** DEFINITION OF VARIABLES ******************
 1540 !*********************************************************
 1550 ! Name$          ! NAME OF SERIAL FILE CREATED TO RECEIVE DATA
 1560 ! Job$           ! DESCRIPTIVE JOB LABEL OF CONTAINED DATA
 1570 ! Medium$        ! ADDRESS OF MASS STORAGE MEDIUM
 1580 ! N_data         ! NUMBER OF DATA ELEMENTS IN EACH VECTOR .
 1590 !*********************************************************
 1600 !********************************
```

```
1610 ! ASSIGN BUFFER I/O PATH TO FILE *
1620 !*******************************
1630 IF Medium$=":INTERNAL" THEN
1640    File_name$=Name$&Medium$
1650 ELSE
1660    File_name$=Medium$&Name$
1670 END IF
1680 ASSIGN @Path_1 TO File_name$
1690 !*******************************
1700 !******* READ  JOB LABEL *********
1710 !*******************************
1720 ENTER @Path_1;Job$
1730 !*******************************
1740 !*** ENTER NUMBER OF ELEMENTS ****
1750 !*******************************
1760 ENTER @Path_1;N_data
1770 !*******************************
1780 !** CORRECTLY SIZE DATA VECTOR ***
1790 !*******************************
1800 REDIM X(N_data-1),Y(N_data-1),Z(N_data-1)
1810 !*******************************
1820 !******** READ  DATA ARRAY ********
1830 !*******************************
1840 ENTER @Path_1;X(*),Y(*),Z(*)
1850 !*******************************
1860 !****** CLOSE FILE AND BUFFER *****
1870 !*******************************
1880 ASSIGN @Path_1 TO *
1890 SUBEND
1900 !***************************************************************
1910 !*********************** SUBROUTINE PLOT_FILE *******************
1920 !***************************************************************
1930 !*      THIS SUBROUTINE ACCEPTS TWO DATA VECTORS AND PLOTS ONE  VERSUS *
1940 !* THE OTHER . THE  USER  NEED  ONLY  SUPPLY  THE  LIMITS OF THE GIVEN *
1950 !* VECTORS AND THE DESIRED PLOTTING COLOR . SCALING AND AXES ARE AUTO- *
1960 !* MATICALLY PROVIDED BY THIS SUBROUTINE . '                          *
1970 !***************************************************************
1980 SUB Plot_file(Xdata(*),Ydata(*),Nplot,Xmin,Xmax,Ymin,Ymax,Penc,New$)
1990 COM /Plot_block/ Xscale,Yscale,Xoffset,Yoffset
2000 !*****************************************************
2010 !*************** DEFINTITION OF LOCAL VARIABLES *************
2020 !*****************************************************
2030 ! Xdata(*)         ! ABSCISSA DATA VECTOR TO BE PLOTTED .
2040 ! Ydata(*)         ! ORDINATE DATA VECTOR TO BE PLOTTED .
2050 ! Nplot            ! NUMBER OF DATA POINTS IN VECTORS .
2060 ! Xmin             ! SMALLEST ELEMENT IN Xdata(*) VECTOR .
2070 ! Xmax             ! LARGEST ELEMENT IN Xdata(*) VECTOR .
2080 ! Ymin             ! SMALLEST ELEMENT IN Ydata(*) VECTOR .
2090 ! Ymax             ! LARGEST ELEMENT IN Ydata(*) VECTOR .
2100 ! Penc             ! DESIRED COLOR CODE OF PLOTTING COLOR  .
2110 ! New$             ! ORDERS THE ROUTINE TO CLEAR THE GRAPHICS
2120 White=1           ! DEFINE THE COLOR CODE FOR WHITE
2130 A_color=White     ! SET AXIS COLOR WHITE
2140 Xleft=0           ! DEFINE LEFT OF SCREEN     (Plotter Units)
2150 Xrail=20          ! DEFINE X AXIS RAIL        (Plotter Units)
2160 Xcenter=64        ! X COORD CENTER SCREEN     (Plotter Units)
2170 Xright=128        ! DEFINE RIGHT SCREEN       (Plotter Units)
2180 Ybottom=0         ! DEFINE LOWER SCREEN       (Plotter Units)
2190 Yrail=16          ! DEFINE Y AXIS RAIL        (Plotter Units)
2200 Ycenter=48        ! Y COORDCENTER SCREEN      (Plotter Units)
2210 Ytop=96           ! DEFINE TOP OF SCREEN      (Plotter Units)
2220 ! X_denom         ! DENOMINATOR OF X PLOTTING SCALE FACTOR .
2230 ! Y_denom         ! DENOMINATOR OF Y PLOTTING  SCALE FACTOR
2240 !*******************************************************
2250 !**********************************************
2260 !** CLEAR AND INITIALIZE GRAPHICS IF SPECIFIED *
```

```
2270 !*************************************************
2280 IF New$="Y" THEN
2290   GINIT 1.5
2300   GRAPHICS ON
2310   PEN White
2320   VIEWPORT Xleft,Xright,Ybottom,Ytop
2330   FRAME
2340   !**********************************
2350   !* DRAW PROPER AXES FOR PLOTTING *
2360   !**********************************
2370   IF Xmin<0 THEN
2380      IF Ymin<0 THEN         !******************************
2390         Xoffset=Xcenter     !*  FOUR QUAD AXES DRAWN HERE *
2400         Yoffset=Ycenter     !******************************
2410         X_denom=Xmax
2420         Y_denom=Ymax
2430         CALL Axis_draw(Xleft,Yoffset,Xright,Yoffset,A_color,-Xmax,Xmax)
2440         CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,-Ymax,Ymax)
2450      ELSE                   !******************************
2460         Xoffset=Xcenter     !* +/- X TYPE AXIS DRAWN HERE *
2470         Yoffset=Yrail       !******************************
2480         X_denom=Xmax
2490         Y_denom=Ymax-Ymin
2500         CALL Axis_draw(Xleft,Yoffset,Xright,Yoffset,A_color,-Xmax,Xmax)
2510         CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,Ymin,Ymax)
2520      END IF
2530   ELSE
2540      IF Ymin<0 THEN        !******************************
2550         Xoffset=Xrail      !* +/- Y TYPE AXIS DRAWN HERE *
2560         Yoffset=Ycenter    !******************************
2570         X_denom=Xmax-Xmin
2580         Y_denom=Ymax-Ymin
2590         CALL Axis_dr w(Xoffset,Yoffset,Xright,Yoffset,A_color,Xmin,Xmax)
2600         CALL Axis_draw(Xoffset,Ybottom,Xoffset,Ytop,A_color,-Ymax,Ymax)
2610         Yoffset=Ybottom
2620      ELSE                  !******************************
2630         Xoffset=Xrail      !* + ONLY X&Y AXES DRAWN HERE *
2640         Yoffset=Ybottom    !******************************
2650         X_denom=Xmax-Xmin
2660         Y_denom=Ymax-Ymin
2670         CALL Axis_draw(Xoffset,Yoffset,Xright,Yoffset,A_color,Xmin,Xmax)
2680         CALL Axis_draw(Xoffset,Yoffset,Xoffset,Ytop,A_color,Ymin,Ymax)
2690      END IF
2700   END IF
2710   Xscale=(Xright-Xoffset)/X_denom
2720   Yscale=(Ytop-Yoffset)/Y_denom
2730 END IF
2740 !*****************************
2750 !* DATA VECTORS PLOTTED BELOW *
2760 !*****************************
2770 PENUP
2780 CALL Scaler(Xdata(0),Ydata(0),Xmin,Xmax,Ymin,Ymax,X_plot,Y_plot)
2790 PEN Penc
2800 MOVE X_plot,Y_plot
2810 FOR I=0 TO Nplot-1
2820   CALL Scaler(Xdata(I),Ydata(I),Xmin,Xma     n,Ymax,X_plot,Y_plot)
2830   DRAW X_plot,Y_plot
2840 NEXT I
2850 SUBEND
2860 !********************************************************************
2870 !*********************** SUBROUTINE AXIS_DRAW *************************
2880 !********************************************************************
2890 !*        THIS SUBROUTINE DRAWS AN AXIS FROM THE STARTING COORDINATE TO *
2900 !* THE FINAL ONE . IT ALSO QUANTIFIES THE ORIGIN AND TERMINUS OF  SAID *
2910 !* AXIS .                                                              *
2920 !********************************************************************
```

```
2930 SUB Axis_draw(Xstart,Ystart,Xfinal,Yfinal,Axis_color,A_min,A_max)
2940 Pie=4*ATN(1)
2950 Delta=5
2960 PENUP
2970 PEN Axis_color
2980 PENUP
2990 MOVE Xstart,Ystart
3000 DRAW Xfinal,Yfinal
3010 PENUP
3020 CSIZE 3.0,.5
3030 CALL Rounder(A_min,3,A0)
3040 CALL Rounder(A_max,3,A1)
3050 IF Xstart=Xfinal THEN
3060   CALL Labelit(Xstart-Delta,Ystart,Pie/2,Axis_color,VAL$(A0))
3070   CALL Labelit(Xfinal-Delta,Yfinal-2*Delta,Pie/2,Axis_color,VAL$(A1))
3080 ELSE
3090   CALL Labelit(Xstart,Ystart-Delta,0,Axis_color,VAL$(A0))
3100   CALL Labelit(Xfinal-2*Delta,Ystart-Delta,0,Axis_color,VAL$(A1))
3110 END IF
3120 SUBEND
3130 !*****************************************************************
3140 !********************** SUBROUTINE LABELIT ***********************
3150 !*****************************************************************
3160 !* THIS SUBROUTINE SIMPLY ACCEPTS THE GIVEN LABEL AND PLACES IT WHERE  *
3170 !* IT IS SPECIFIED (ie X,Y LOCATION) AT THE GIVEN TILT ANGLE . THE PEN  *
3180 !* COLOR 'Penc' IS ALSO  PROVIDED  BY  THE  USER  . THIS SAVES A LOT OF *
3190 !* REPETITIVE CODE .                                                   *
3200 !*****************************************************************
3210 SUB Labelit(X,Y,Tilt,Penc,Strng$)
3220 PENUP
3230 MOVE X,Y
3240 PEN Penc
3250 LDIR Tilt
3260 LABEL Strng$
3270 PENUP
3280 SUBEND
3290 !*****************************************************************
3300 !************************* SUBROUTINE SCALER *********************
3310 !*****************************************************************
3320 !*      THIS SUBROUTINE SCALES THE DATA PASSED TO IT FOR CRT PLOTTING  *
3330 !* PURPOSES .                                                          *
3340 !*****************************************************************
3350 SUB Scaler(X_data,Y_data,Xmin,Xmax,Ymin,Ymax,X_plot,Y_plot)
3360 COM /Plot_block/ Xscale,Yscale,Xoffset,Yoffset
3370 X_plot=Xscale*(X_data-Xmin)+Xoffset
3380 Y_plot=Yscale*(Y_data-Ymin)+Yoffset
3390 SUBEND
3400 !*****************************************************************
3410 !*********************** SUBROUTINE ROUNDER **********************
3420 !*****************************************************************
3430 !*       THIS  SUBROUTINE  ACCEPTS  A  NUMBER  OF ANY SIZE OR SIGN AND *
3440 !* ROUNDS IT TO THE SPECIFIED NUMBER OF DIGITS .                       *
3450 !*****************************************************************
3460 SUB Rounder(X_input,N_digits,X_rounded)
3470 !**************************************************
3480 !************ DEFINITION OF LOCAL VARIABLES ***************
3490 !**************************************************
3500 ! X_input       ! INPUT NUMBER TO BE ROUNDED
3510 ! X_dummy       ! DUMMY VARIABLE USED TO PROTECT X_input
3520 ! N_digits      ! NUMBER OF DIGITS DISPLAYED AFTER ROUNDING
3530 ! X_rounded     ! ROUNDED EQUIVALENT OF X_input
3540 ! Sign          ! NUMERICAL POLARITY OF ROUNDED NUMBER
3550 ! Magnitude     ! ORDER OF MAGNITUDE OF INPUT NUMBER
3560 ! Mantissa      ! MANTISSA OF NUMBER UNDER ROUNDING
3570 ! ARGUMENT      ! ABBREVIATED VERSION OF MANTISSA.
3580 !**************************************************
3590 IF X_input<>0 THEN
3600   X_dummy=X_input
3610   Sign=SGN(X_dummy)
3620   X_dummy=ABS(X_dummy)
3630   Magnitude=INT(LGT(X_dummy))
3640   Mantissa=X_dummy/(10^Magnitude)
3650   Argument=INT(Mantissa*10^(N_digits-1))/10^(N_digits-1)
3660   X_rounded=Sign*Argument*10^Magnitude
3670 ELSE
3680   X_rounded=X_input
3690 END IF
3700 SUBEND
```